Online Mixed Packing and Covering *

Umang Bhaskar† Lisa Fleischer†

March 1, 2013

Abstract

In many problems, the inputs to the problem arrive over time. As each input is received, it must be dealt with irrevocably. Such problems are *online* problems. An increasingly common method of solving online problems is to solve the corresponding linear program, obtained either directly for the problem or by relaxing the integrality constraints. If required, the fractional solution obtained is then rounded online to obtain an integral solution.

We give algorithms for solving linear programs with mixed packing and covering constraints online. We first consider mixed packing and covering linear programs, where packing constraints $\mathbf{Px} \leq \mathbf{p}$ are given offline and covering constraints $\mathbf{Cx} \geq \mathbf{c}$ are received online. The objective is to minimize the maximum multiplicative factor by which any packing constraint is violated, while satisfying the covering constraints. For general mixed packing and covering linear programs, no prior sublinear competitive algorithms are known. We give the first such — a polylogarithmic-competitive algorithm for solving mixed packing and covering linear programs online. We also show a nearly tight lower bound.

Our techniques for the upper bound use an exponential penalty function in conjunction with multiplicative updates. While exponential penalty functions are used previously to solve linear programs offline approximately, offline algorithms know the constraints beforehand and can optimize greedily. In contrast, when constraints arrive online, updates need to be more complex.

We apply our techniques to solve two online fixed-charge problems with congestion. These problems are motivated by applications in machine scheduling and facility location. The linear program for these problems is more complicated than mixed packing and covering, and presents unique challenges. We show that our techniques combined with a randomized rounding procedure can be used to obtain polylogarithmic-competitive integral solutions. These problems generalize online set-cover, for which there is a polylogarithmic lower bound. Hence, our results are close to tight.

†Dartmouth College, 6211 Sudikoff Lab, Hanover NH 03755. {umang, lkf}@cs.dartmouth.edu.

*This work was supported in part by NSF grants CCF-0728869 and CCF-1016778.

1 Introduction

In this paper, we give the first online algorithm for general mixed packing and covering linear programs (LPs) with a sublinear competitive ratio. The problem we study is as follows.

Online Mixed Packing and Covering (OMPC). Given: a set of *packing constraints* $\mathbf{Px} \leq \mathbf{p}$, and a set of *covering constraints* $\mathbf{Cx} \geq \mathbf{c}$, with positive coefficients and variables $\mathbf{x}$, such that the packing constraints are known in advance, and the covering constraints arrive one at a time. Goal: After the arrival of each covering constraint, increase $\mathbf{x}$ so that the new covering constraint is satisfied and the amount λ by which we must multiply $\mathbf{p}$ to make the packing constraints feasible is as small as possible.

Mixed packing and covering problems model a wide range of problems in combinatorial optimization and operations research. These problems include facility location, machine scheduling, and circuit routing. In these problems, requests for resources such as bandwidth or processing time are received over time, or *online*, whereas the set of resources is known offline. As each request arrives, we must allocate resources to satisfy the request. These allocations are often impossible or costly to revoke. The resources correspond to packing constraints in our setting and are known offline. Requests correspond to covering constraints, and arrive online. The performance of an online algorithm is measured by the *competitive ratio*, defined as the worst case ratio of the value of the solution obtained by the online algorithm to the value obtained by the optimal offline algorithm which has as its input the entire sequence of requests. The worst case ratio is over all possible sequences of inputs.

Many techniques to solve integer problems online first obtain a fractional solution, and then round this to an integer solution [1, 2, 5, 6]. The first step involves solving a linear program relaxation of the original problem online. In fact, this can be a bottleneck step in obtaining a good competitive ratio. Thus, our algorithm for online mixed packing and covering can provide an important first step in obtaining good online solutions to several combinatorial problems. We demonstrate the power of our ideas by extending them to give the first online algorithms with sublinear competitive ratios for a number of fixed-charge problems with capacity constraints. For these problems, we first solve the linear program relaxation online, and then use known randomized rounding techniques to obtain an integer solution online.

Applications. We use our techniques to study two problems with fixed-charge and congestion costs. Both fixed-charge problems and congestion problems are widely studied offline and online; we discuss specific applications and references below. In general, fixed charges are used to model one-time costs such as resource purchases or installation costs, while congestion captures the load on any resource. In machine scheduling, for example, the makespan can be modelled as the maximum congestion by setting each resource to be a machine and setting unit capacity for each machine.

Application 1: Unrelated Machine Scheduling with Start-up Costs (UMSC). Given offline: a set of machines $\{1, \ldots, m\}$ with start-up cost c_i for machine i. Jobs arrive online, and job j requires p_{ij} time to be processed on machine i. Goal: when a job j arrives, determine whether to "open" new machines by paying their start-up cost, and then assign the job to one of the open machines, so that the sum of the *makespan* of the schedule — the maximimum over machines of the processing times of the jobs assigned to it — and the sum of start-up costs is minimized.

The problem of scheduling jobs to minimize the makespan and the fixed charges is studied both offline [17, 26] and online [14, 25, 24]. The problem is motivated by reducing energy consumption in large data centers, such as those used by Google and Amazon [8, 26]. The energy consumption of a large data center is estimated to exceed that of thousands of homes, and the energy costs of these centers is in the tens of millions of dollars [32], hence algorithms that focus on reducing energy consumption are of practical importance. The inclusion of a fixed charge models the cost of starting up a machine. Thus machines do not need to stay on, and can be started when the load increases. Bicriteria results for the offline problem are

given in [26] and [17] using different techniques. We show strong lower bounds for bicriteria results in the online setting, and therefore focus on algorithms for the sum objective. For the online problem with identical machines, [14, 25] give constant-competitive algorithms for the sum objective. These are extended to the case where machines have speed either 1 or s, with more general costs for the machines in [24].

Application 2: Capacity Constrained Facility Location (CCFL). Given offline: a set of facilities $\mathcal{F}$ with fixed-charge c_i and capacity u_i for each facility i in $\mathcal{F}$. Clients arrive online, and each client j has an assignment cost a_{ij} and a demand p_{ij} on being assigned to facility i. Goal: when client j arrives, determine whether to open new facilities by paying their fixed charge, and then assign the client to an open facility, so that the sum of the maximum congestion of any facility, total assignment costs, and the total fixed charges paid for opened facilities is minimized. The congestion of a facility is the ratio of the sum of the loads of clients assigned to the facility to the capacity of the facility.

For online facility location without capacity constraints, a $\Theta\left(\frac{\log n}{\log\log n}\right)$-competitive ratio is possible when the assignment costs form a metric [18], and a $O(\log m \log n)$-competitive ratio is possible when assignment costs are non-metric [1], with n clients and m facilities. Capacitated facility location is a natural extension to the problem. In the offline setting, constant-factor approximation algorithms are known for both facility location with soft capacities — when multiple facilities can be opened at a location — and hard capacities — when either a single facility or no facility is opened at each location [30, 35]. Our problem is a variant of non-metric soft-capacitated facility location where instead of minimizing the cost of installing multiple facilities at a location, we minimize the load on the single facility at each location, in addition to fixed-charge and assignment costs.

Our Results. We give polylogarithmic competitive ratios for the problems discussed. Our results are the first sublinear guarantees for these problems.

- For OMPC:
  - A deterministic $O(\ln m \ln(d\rho\kappa))$-competitive algorithm, where m is the number of packing constraints, d is the maximum number of variables in any constraint, ρ is the ratio of the maximum to the minimum non-zero packing coefficient and κ is the ratio of the maximum to the minimum non-zero covering coefficient (Section 2). If all coefficients are either 0 or 1, this gives a $O(\ln m \ln d)$-competitive algorithm.
  - A lower bound of $O(\ln m \ln(d/\ln m))$ for any deterministic algorithm for OMPC. Our algorithm for OMPC is thus nearly tight (Section 2.3).
- For CCFL and UMSC:
  - A randomized $O(\ln(mn\rho)\ln^2(mn))$-competitive algorithm for CCFL, where m and n are the number of facilities and clients respectively, and ρ is is the ratio of the maximum to the minimum total cost of assigning a single client (Section 3). We obtain the same competitive ratio for UMSC, where m and n are the number of machines and jobs respectively, and ρ is the ratio of the maximum to the minimum total cost of assigning a single job.
  - A lower bound for bicriteria results for CCFL: even if the maximum congestion T is given offline, no deterministic online algorithm can obtain a fractional solution with maximum congestion $o(m)T$ and fixed-charge within a polylogarithmic factor of the optimal (Section 3.5). This lower bound also holds for UMSC , where T is the makespan.

Since each of our applications include fixed-charges as part of the objective, they generalize online set cover. In UMSC, for example, set cover is obtained by setting the processing times to be either zero or

infinity. Since the makespan in any bounded solution to the problem is now zero, this reduces the problem to covering jobs with machines to minimize the sum of machine startup costs. Online set cover has a lower bound of $\Omega(\log m \log n)$ on the competitive ratio assuming BPP $\neq$ NP [3]. Thus, our results for UMSC and CCFL are tight modulo a logarithmic factor.

Our Techniques. Our techniques for online mixed packing and covering are based on a novel extension of multiplicative weight updates. We replace the packing constraints in our problem with an exponential penalty function that gives an upper bound on the violation of any constraint. When a covering constraint arrives, the increment to any variable is inversely proportional to the rate of change of this penalty function with respect to the variable. We use a primal-dual analysis to show that this technique, combined with a doubling approach used in previous online algorithms, gives the required competitive ratio. While exponential potential functions are widely used for offline algorithms and machine learning, e.g. [4], our work is the first to use an exponential potential function to drive multiplicative updates that yield provably good competitive ratios for *online* algorithms.

Our work is closely related to work on solving pure packing packing and pure covering linear programs online, and Lagrangean-relaxation techniques for solving linear programs approximately offline. Multiplicative weight updates are used in [10] to obtain $O(\log n)$-competitive fractional solutions for covering linear programs when the constraints arrive online. In [10], the cost is a simple linear function of the variables. The update to each variable is inversely proportional to the sensitivity of the cost function relative to the variable, given by the variable's coefficient in the cost function. In our problem, however, the cost is the maximum violation of any packing constraint. The cost function is thus nonlinear, and since its sensitivity relative to a variable changes, it is not apparent how to extend the techniques from [10]. We use an exponential potential function to obtain a differentiable approximation to this nonlinear cost. For each variable, our updates depend on the sensitivity of this potential function relative to the variable. In addition to the primal-dual techniques in [10], a key step in our analysis is to obtain bounds on the rate of change of this potential function.

A large body of work uses Lagrangean-relaxation techniques to obtain approximate algorithms for solving LPs offline, e.g., [31, 34]. In these papers, the constraints in the LP are replaced by an exponential penalty function. In each update, the update vector for the variables minimizes the change in the penalty function. In this sense, the updates in these offline algorithms are greedy. Since the constraints are available offline, this gives ϵ-approximate solutions. In our case, since covering constraints arrive online, greedy algorithms perform very poorly, and we must use different techniques. We use an exponential penalty function similar to offline algorithms. However, our updates are very different. Instead of a greedy strategy as used in [31, 34], we hedge our bets and increment all variables that appear in the covering constraint. The increment to each variable is inversely proportional to its contribution to the penalty function.

For fixed-charge problems with capacity constraints, we solve the corresponding linear programs online and, for our applications, round the fractional solutions to obtain integral solutions online. The linear programs for these problems are significantly more complicated than mixed packing and covering. We combine our techniques for mixed packing and covering with a more complex doubling approach to obtain fractional solutions, and adapt randomized rounding procedures used previously offline for machine scheduling [26] and online for set cover [11] to obtain integral solutions.

Other Related Work. Multiplicative updates are used in a wide variety of contexts. They are used in both offline approximation algorithms for packing and covering problems [7, 16, 19, 20, 21, 22, 27, 28, 29, 31, 33, 34], and online approximations for pure packing or pure covering problems based on linear programs such as set cover [12], caching [6], paging [5], and ad allocations [9]. Both offline and online, these algorithms are analyzed using a primal-dual framework. Multiplicative updates are used earlier [1, 2] to implicitly solve a linear program online for various network optimization problems. The fractional solution

obtained was rounded online to obtain an integral solution. Multiplicative weight updates also have a long history in learning theory; these results are surveyed in [4].

Our work studies the worst-case behaviour of our algorithms assuming adversarial inputs. A large body of work studies algorithms for online problems when the inputs are received as the result of a stochastic process. Two common models studied in the literature are (1) when the inputs are picked from a distribution (either known or unknown), and (2) when an adversary picks the inputs, but the inputs are presented to the algorithm in random order. The adwords and display ads problems can be modeled as packing linear programs with variables arriving online. A number of papers give algorithms for these problems assuming stochastic inputs; some of these results are presented in [13, 15].

2 Online Mixed Packing and Covering

In this section, we consider mixed packing and covering linear programs. A mixed packing and covering linear program has two types of constraints: covering constraints of the form $\mathbf{Cx} \geq \mathbf{c}$, and packing constraints of the form $\mathbf{Px} \leq \mathbf{p}$. We normalize the constraints so that the right side of each constraint is 1. Our objective is to obtain a solution $\mathbf{x}$ that minimizes the maximum amount by which any packing constraint is violated. Thus, our problem is to obtain a solution to the following linear program:

$$\min \lambda \text{ s.t. } \mathbf{Cx} \geq \mathbf{1},\ \mathbf{Px} \leq \lambda,\ \mathbf{x}, \lambda \geq \mathbf{0}\,. \tag{1}$$

The packing constraints are given to us initially, and the covering constraints are revealed one at a time. Our online algorithm assigns fractional values to the variables. As covering constraints arrive, the variable values can be increased, but cannot be decreased.

For a vector $\mathbf{v}$, we use both v_i and $(v)_i$ to denote its ith component. We use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. The vector of all ones and all zeros is denoted by $\mathbf{1}$ and $\mathbf{0}$, respectively.

The number of variables, number of packing constraints, and number of covering constraints in the linear program are denoted by n, m, and m_c respectively. We use d to denote the maximum number of variables in any constraint. We define $\rho := \max_{k,j} p_{kj} / \min_{k,j:p_{kj}>0} p_{kj}$ and similarly $\kappa = \max_{i,j} c_{ij} / \min_{i,j:c_{ij}>0} c_{ij}$. The value of κ is used only in the analysis of the algorithm; we do not need to know its value during execution. Define $\kappa_1 := \max_j c_{1j}$, i.e., κ_1 is the maximum coefficient in the first covering constraint to arrive. Define d_1 as the maximum number of variables in any packing constraint, and the first covering constraint. Define $\mu := 1 + \frac{1}{3\ln(em)}$, and $\sigma := e^2 \ln(\mu d^2 \rho \kappa)$. Here, e is the base of the natural logarithm.

We use OPT to denote the optimal value of λ given $\mathbf{P}$ and $\mathbf{C}$, hence OPT is the value returned by the optimal offline algorithm.

In order to analyze our algorithm, we consider the dual of (1) as well:

$$\max \sum_i y_i \text{ s.t. } \mathbf{C^T y} \leq \mathbf{P^T z},\ \ \sum_{k=1}^{m} z_k \leq 1,\ \ \mathbf{y}, \mathbf{z} \geq \mathbf{0} \tag{2}$$

2.1 An Algorithm for Mixed Packing and Covering Online

We now give an algorithm for solving OMPC and show that it is $O(\log(d\rho\kappa)\log m)$-competitive. We assume in the following discussion that we are given a scaling parameter $\Gamma \geq \max_{k,j} p_{kj}/(d_1 \rho \kappa_1)$, which is used to scale the matrix of packing coefficients $\mathbf{P}$. In Theorem 11, we show that if $2OPT \geq \frac{\Gamma}{4\sigma} \geq OPT$ then our algorithm yields the stated competitive ratio. Without this estimate Γ, we can use a "doubling procedure" commonly used in online algorithms, which increases the competitive ratio obtained by a factor of 4 (Section 2.2).

Given a vector $\mathbf{x}$, let $\lambda(\mathbf{x}) := \max_{k\in[m]}(\mathbf{P}\mathbf{x})_k$. For a given scaling parameter Γ, let $\tilde{\mathbf{P}} := \mathbf{P}/\Gamma$, $\tilde{p}_{kj} := p_{kj}/\Gamma$ and $\tilde{\lambda}(\mathbf{x}) := \max_{k\in[m]}(\tilde{\mathbf{P}}\mathbf{x})_k$. Let $\mathrm{est}(\mathbf{x}) := \ln\left(\sum_{k\in[m]}\exp(\tilde{\mathbf{P}}\mathbf{x})_k\right)$ be an estimate of $\tilde{\lambda}(\mathbf{x})$, and note that $\max_k(\tilde{\mathbf{P}}\mathbf{x})_k \le \mathrm{est}(\mathbf{x}) \le \max_k(\tilde{\mathbf{P}}\mathbf{x})_k + \ln m$. For each variable x_j, define

$$\mathrm{rate}_j(\mathbf{x}) := \frac{\partial \mathrm{est}(\mathbf{x})}{\partial x_j} = \frac{\sum_{k\in[m]}\tilde{p}_{kj}\exp(\tilde{\mathbf{P}}\mathbf{x})_k}{\sum_{k\in[m]}\exp(\tilde{\mathbf{P}}\mathbf{x})_k}\,. \tag{3}$$

Our algorithm is given as Algorithm 1. Upon receiving the first constraint, we initialize $x_j \leftarrow 1/(d_1^2\rho\kappa_1)$ for all $j \in [n]$. We also initialize a counter variable $l \leftarrow 0$.

When a covering constraint $(\mathbf{C}\mathbf{x})_i \ge 1$ arrives it gets assigned a new dual variable y_i, and the variables are incremented as described. The dual variables $\mathbf{y}$ are used only in the analysis.

For covering constraint i, define

$$\epsilon_i(\mathbf{x}) := (\mu-1)\min_{j:c_{ij}>0}\mathrm{rate}_j(\mathbf{x})/c_{ij}\,, \tag{4}$$

so that for all $j \in [n]$, $\epsilon_i(\mathbf{x})c_{ij}/\mathrm{rate}_j(\mathbf{x}) \le \mu - 1$. In line 8, each variable x_j gets increased by at most a factor of μ, and at least one variable gets incremented by a factor of μ.

Algorithm 1 MPC-APPROX: Upon arrival of ith covering constraint:

1: When first constraint arrives, initialize $x_j \leftarrow 1/(d_1^2\rho\kappa_1)$ for all $j \in [n]$, and $l \leftarrow 0$.
2: Upon arrival of ith covering constraint:
3: **while** $(\mathbf{C}\mathbf{x})_i < 1$ **do**
4: $\quad l \leftarrow l+1$, $\mathbf{x}^l \leftarrow x$
5: $\quad \forall j$, $\mathrm{rate}_j \leftarrow \mathrm{rate}_j(\mathbf{x}^l)$ /* defined in (3) */
6: $\quad \epsilon_i \leftarrow \epsilon_i(\mathbf{x}^l)$ /* defined in (4) */
7: $\quad$ **for** $j \in [n]$ **do**
8: $\quad\quad x_j \leftarrow x_j\left(1+\epsilon_i\frac{c_{ij}}{\mathrm{rate}_j}\right)$
9: $\quad y_i \leftarrow y_i + e\epsilon_i$ /* for analysis */
10: $\quad$ **if** $\tilde{\lambda}(\mathbf{x}) \ge 3\ln(em)$ **then return** FAIL

A single iteration of the while loop is a *phase*, indexed by l, and the first phase is phase 0. The value of the variables before they are incremented in phase l is $\mathbf{x}^l$. $\mathbf{x}^0$ denotes the values after initialization. For covering constraint i, L_i is the indices of the phases executed from its arrival until $(\mathbf{C}\mathbf{x})_i \ge 1$, and $L := \cup_i L_i$.

We first show upper bounds on values attained by the variables, and on the running time.

Lemma 1. *During the execution of the algorithm, for any $j \in [n]$, $x_j \le \mu/\min_{i:c_{ij}>0} c_{ij}$.*

Proof. For any x_j, if $\min_{i:c_{ij}>0} c_{ij}x_j \ge 1$, then x_j will not be incremented further in any phase since any covering constraint i with $c_{ij} > 0$ must already be satisfied. Thus, since the value of any variable increases by at most a factor of μ in a phase, $x_j \le \mu/\min_{i:c_{ij}>0} c_{ij}$. □

Lemma 2. MPC-APPROX *executes $O(n\ln(\mu d^2\rho\kappa)\ln m)$ phases, and each phase takes time $O(mn)$.*

Proof. In each phase, the value of at least one variable gets incremented by a factor of μ. Each variable has an initial value of $1/(d_1^2\rho\kappa_1)$. Let n_j be the number of phases in which x_j gets increased by a factor of μ. Then $x_j \ge \mu^{n_j}/(d_1^2\rho\kappa_1)$. Since by Lemma 1, $x_j \le \mu/\min_{i:c_{ij}>0} c_{ij}$, $n_j \le \log_\mu(\mu d_1^2\rho\kappa_1/\min_{i:c_{ij}>0} c_{ij})$. Observing that for all j, $\kappa_1/\min_{i:c_{ij}>0} c_{ij} \le \kappa$ and $d_1 \le d$, it follows that each variable can be increased

by μ in at most $\log_\mu(\mu d^2\rho\kappa)$ phases. Since in each phase at least one variable increases by a factor of μ, the number of phases is at most $n\log_\mu(\mu d^2\rho\kappa) = n\ln(\mu d^2\rho\kappa)/\ln\mu$. Since $\ln(1+x) \geq x/e$ for $0 \leq x \leq 1$, $\ln\mu \geq 1/(3e\ln(em))$. Thus the number of phases is at most $O(n\log(\mu d^2\rho\kappa)\log m)$. In each phase, $\tilde{\mathbf{P}}\mathbf{x}$ can be computed in $O(mn)$ time; then $\mathrm{est}(\mathbf{x})$ and each $\mathrm{rate}_j(\mathbf{x})$ can be computed in time $O(m)$. Thus each phase takes time $O(mn)$. □

Our proof of the competitive ratio follows from a primal-dual analysis. We show in Corollary 6 that $\ln m + OPT/\Gamma$ plus the value of the dual objective maintained by the algorithm is an upper bound on the primal objective maintained by the algorithm. Lemmas 7 and 8 show how the dual variables maintained by the algorithm can be scaled down to obtain feasible dual values. We show in Theorem 11 that together these prove the bound on the competitive ratio.

We first show that the initialization of the variables ensures that $\mathrm{est}(\mathbf{x}^0)$ does not exceed $\frac{OPT}{\Gamma} + \ln m$.

Lemma 3. *For the variables as initialized, $\tilde{\lambda}(\mathbf{x}^0) \leq OPT/\Gamma$, and hence $\mathrm{est}(\mathbf{x}^0) \leq \frac{OPT}{\Gamma} + \ln m$.*

Proof. Let x_j^* be the values for the variables in an optimal solution. After the first covering constraint is received, $1 \leq \sum_j c_{1j}x_j^* \leq \max_r c_{1r}\sum_j x_j^*$. Since the first covering constraint has at most d_1 variables, there exists variable $x_b^* \geq 1/(d_1\max_r c_{1r})$, and hence

$$OPT \;=\; \max_{k\in[m]}(\mathbf{P}\mathbf{x}^*)_k \;\geq\; \min_{k,j:p_{kj}>0} p_{kj}x_b^* \;\geq\; \min_{k,j:p_{kj}>0} p_{kj}/(d_1\max_r c_{1r}) \;=\; \min_{k,j:p_{kj}>0} p_{kj}/(d_1\kappa_1)\,.$$

Using $\rho = \max_{k,j} p_{kj}/\min_{k,j:p_{kj}>0} p_{kj}$,

$$OPT \;\geq\; \max_{k,j:p_{kj}>0} p_{kj}/(d_1\rho\kappa_1) \;=\; \Gamma \max_{k,j:p_{kj}>0} \tilde{p}_{kj}/(d_1\rho\kappa_1)\,. \tag{5}$$

Our algorithm initializes $x_j^0 = 1/(d_1^2\rho\kappa_1)$, and hence

$$\tilde{\lambda}(\mathbf{x}^0) \;=\; \max_{k\in[m]}(\tilde{\mathbf{P}}\mathbf{x}^0)_k \;\leq\; d_1\max_{k,j}\tilde{p}_{kj}/(\Gamma d_1^2\rho\kappa_1) \;\leq\; \max_{k,j}\tilde{p}_{kj}/(\Gamma d_1\rho\kappa_1) \overset{(5)}{\leq} OPT/\Gamma\,, \tag{6}$$

where the first inequality is because any packing constraint has at most d_1 variables. Thus, $\mathrm{est}(\mathbf{x}^0) \leq \tilde{\lambda}(\mathbf{x}^0) + \ln m \leq (OPT/\Gamma) + \ln m$, proving the lemma. □

Corollary 4. *If $\Gamma \geq \max_{k,j} p_{kj}/(d_1\rho\kappa_1)$, then $\tilde{\lambda}(\mathbf{x}^l) \leq 3\ln(em)$ at the beginning of any phase l.*

Proof. Since $\Gamma \geq \max_{k,j} p_{kj}/(d_1\rho\kappa_1)$, by (6), $\tilde{\lambda}(\mathbf{x}^0) \leq 1$. Thus the lemma is satisfied for the first phase. For any phase $l > 0$, the algorithm would have failed at the end of phase $l-1$ if $\tilde{\lambda}(\mathbf{x}) \geq 3\ln(em)$. Since the algorithm did not fail in phase $l-1$, in any phase l, $\tilde{\lambda}(\mathbf{x}^l) \leq 3\ln(em)$. □

Lemma 5. *If $\Gamma \geq \max_{k,j} p_{kj}/(d_1\rho\kappa_1)$, the increase in the dual objective $\sum_i y_i$ is an upper bound on the increase in $\mathrm{est}(\mathbf{x})$ in every phase.*

Proof. Let est^l and est^{l+1} denote the values of $\mathrm{est}(\mathbf{x})$ before and after the variables are incremented in phase l, respectively. We will show that $\mathrm{est}^{l+1} - \mathrm{est}^l \leq e\epsilon_i$, which is the increase in $\sum_i y_i$ in phase l.

Let $\mathbf{x}^l$ and $\mathbf{x}^{l+1}$ be the values of $\mathbf{x}$ before and after being incremented in phase l. For each x_j, let $g_j(t) := x_j^l + (x_j^{l+1} - x_j^l)t$ for $0 \leq t \leq 1$. Note that $g_j(0) = x_j^l$ and $g_j(1) = x_j^{l+1}$. Define $\mathbf{g}(t) = (g_1(t), g_2(t), \ldots, g_m(t))$. With some abuse of notation, any function of $\mathbf{x}$, say $h(\mathbf{x})$, can be viewed as a function of t, with $h(t) := h(\mathbf{g}(t))$. Thus, the functions $\mathrm{est}(\mathbf{x})$ and $\mathrm{rate}_j(\mathbf{x})$ can be written as functions of t: $\mathrm{est}(t) = \ln\sum_{k\in[m]}\exp(\tilde{\mathbf{P}}\mathbf{g}(t))_k$, and

$$\text{rate}_j(t) \;=\; \text{rate}_j(\mathbf{g}(t)) \;=\; \frac{\partial \text{est}(t)}{\partial g_j(t)} \;=\; \frac{\sum_{k\in[m]} \tilde{p}_{kj} \exp(\tilde{\mathbf{P}}\mathbf{g}(t))_k}{\exp(\text{est}(t))}\,. \tag{7}$$

We use these alternate expressions in the remainder of the proof. By the chain rule,

$$\frac{d\text{est}(t)}{dt} \;=\; \sum_{j=1}^{n} \frac{\partial \text{est}(t)}{\partial g_j(t)} \frac{dg_j(t)}{dt}\,,$$

and hence,

$$\text{est}^{l+1} - \text{est}^{l} \;=\; \int_{t=0}^{1} \frac{d\text{est}(t)}{dt} dt \;=\; \sum_{j=1}^{n} \int_{t=0}^{1} \text{rate}_j(t) \frac{dg_j(t)}{dt} dt\,. \tag{8}$$

In a phase, each variable is incremented by at most a factor of μ. Therefore $\mathbf{x}^{l+1} \le \mu \mathbf{x}^l$. Then since $\mathit{\Gamma} \ge \max_{k,j} p_{kj}/(d_1\rho\kappa_1)$, by Corollary 4, $\tilde{\lambda}(\mathbf{x}^l) \le 3\ln(em)$ in any phase l. Thus $\text{rate}_j(t) \le e\text{rate}_j(0)$ for $0 \le t \le 1$ by Lemma 45 (in Appendix). Hence

$$\text{est}^{l+1} - \text{est}^{l} \;\le\; e\sum_{j=1}^{n} \text{rate}_j(\mathbf{x}^l) \int_{t=0}^{1} \frac{dg_j(t)}{dt} dt \;=\; e\sum_{j=1}^{n} \text{rate}_j(\mathbf{x}^l)(x_j^{l+1} - x_j^l)\,.$$

Since in phase l each variable x_j gets multiplied by $1 + \epsilon_i(\mathbf{x}^l)\frac{c_{ij}}{\text{rate}_j(\mathbf{x}^l)}$,

$$\text{est}^{l+1} - \text{est}^{l} \;\le\; e\epsilon_i(\mathbf{x}^l) \sum_{j=1}^{n} \text{rate}_j(\mathbf{x}^l) \frac{c_{ij}x_j^l}{\text{rate}_j(\mathbf{x}^l)} \;=\; e\epsilon_i(\mathbf{x}^l) \sum_{j=1}^{n} c_{ij}x_j^l \;\le\; e\epsilon_i(\mathbf{x}^l)$$

where the last inequality follows since, on entering the for loop, $(\mathbf{C}\mathbf{x})_i < 1$. Since $e\epsilon_i(\mathbf{x}^l)$ is the increase in the dual objective, this proves the lemma. □

Corollary 6. *If $\mathit{\Gamma} \ge \max_{k,j} p_{kj}/(d_1\rho\kappa_1)$, then $\sum_i y_i \ge \text{est}(\mathbf{x}) - \frac{OPT}{\mathit{\Gamma}} - \ln m$.*

Proof. By Lemma 5, the increase in $\sum_i y_i$ is an upper bound on the increase in $\text{est}(\mathbf{x})$, thus $\sum_i y_i \ge \text{est}(\mathbf{x}) - \text{est}(\mathbf{x}^0)$. By Lemma 3, $\text{est}(\mathbf{x}^0) \le \ln m + \frac{OPT}{\mathit{\Gamma}}$, and hence $\sum_i y_i \ge \text{est}(\mathbf{x}) - \ln m - \frac{OPT}{\mathit{\Gamma}}$. □

We now show that the dual variables do not violate the dual constraints by much. We choose the dual variable z_k corresponding to each packing constraint $k \in [m]$ as

$$z_k \;:=\; \max_{l\in L} \frac{\exp((\tilde{\mathbf{P}}\mathbf{x}^l)_k)}{\exp(\text{est}(\mathbf{x}^l))} \tag{9}$$

Lemma 7. *For $\mathbf{z}$ as defined in (9),* $\sum_{k\in[m]} z_k \;\le\; \ln(em) + \max_{l\in L} \tilde{\lambda}(\mathbf{x}^l)$.

Proof. For each packing constraint k, let $\phi(k) := \arg\max_l \frac{\exp((\tilde{\mathbf{P}}\mathbf{x}^l)_k)}{\exp(\text{est}(\mathbf{x}^l))}$. Thus z_k attains its value in phase $\phi(k)$. We index the packing constraints so that $\phi(1) \le \phi(2) \le \cdots \le \phi(m)$. Then for any $r, k \in [m]$ with $k \ge r$ so that $\phi(k) \ge \phi(r)$, we have $(\tilde{\mathbf{P}}\mathbf{x}^{\phi(k)})_r \ge (\tilde{\mathbf{P}}\mathbf{x}^{\phi(r)})_r$ since the variables x are increasing. Thus,

$$\exp(\text{est}(\mathbf{x}^{\phi(k)})) \;=\; \sum_{r\in[m]} \exp((\tilde{\mathbf{P}}\mathbf{x}^{\phi(k)})_r) \;\ge\; \sum_{r\le k} \exp((\tilde{\mathbf{P}}\mathbf{x}^{\phi(k)})_r) \;\ge\; \sum_{r\le k} \exp((\tilde{\mathbf{P}}\mathbf{x}^{\phi(r)})_r)\,. \tag{10}$$

Substituting (10) into (9) yields

$$z_k \;=\; \frac{\exp((\tilde{\mathbf{P}}\mathbf{x}^{\phi(k)})_k)}{\exp(\mathrm{est}(\mathbf{x}^{\phi(k)}))} \;\le\; \frac{\exp((\tilde{\mathbf{P}}\mathbf{x}^{\phi(k)})_k)}{\sum_{r\le k}\exp(\tilde{\mathbf{P}}\mathbf{x}^{\phi(r)})_r}\,.$$

Then by Lemma 44 in the appendix, with $a_k = \exp((\tilde{\mathbf{P}}\mathbf{x}^{\phi(k)})_k)$,

$$\sum_{k\in[m]} z_k \;\le\; 1+\ln\left(\frac{\sum_{k\in[m]}\exp((\tilde{\mathbf{P}}\mathbf{x}^{\phi(k)})_k)}{\exp(\tilde{\mathbf{P}}\mathbf{x}^{\phi(1)})_1}\right) \;\le\; 1+\ln m+\max_l \tilde{\lambda}(\mathbf{x}^l)\,, \tag{11}$$

where the last inequality follows since $\exp((\tilde{\mathbf{P}}\mathbf{x}^{\phi(1)})_1)\ge 1$ and $(\tilde{\mathbf{P}}\mathbf{x}^l)_k \le \tilde{\lambda}(\mathbf{x}^l)$ by definition of $\tilde{\lambda}$. □

The next lemma tells us how much we must scale the dual solution obtained by the algorithm to obtain a dual feasible solution.

Lemma 8. *For any $j\in[n]$, $(\mathbf{C^T y})_j \;\le\; (\mathbf{P^T z})_j\,\dfrac{\sigma}{\Gamma}$.*

Proof. Consider a phase l executed upon arrival of a covering constraint i. In this phase, y_i gets incremented by $e\epsilon_i(\mathbf{x}^l)$. This increment occurs in every phase in L_i. Hence

$$(\mathbf{C^T y})_j \;=\; \sum_{i\in[m_c]} c_{ij}y_i \;=\; e\sum_{i\in[m_c]} c_{ij}\sum_{l\in L_i}\epsilon_i(\mathbf{x}^l)\,. \tag{12}$$

By Lemma 1, $x_j \le \mu/\min_{i:c_{ij}>0} c_{ij}$. Further, since the initial value of x_j is $1/(d_1^2\rho\kappa_1)$ and is multiplied by $\left(1+\epsilon_i\frac{c_{ij}}{\mathrm{rate}_j}\right)$ in every phase, for all $j\in[n]$,

$$\frac{\mu}{\min_{i:c_{ij}>0} c_{ij}} \;\ge\; x_j \;=\; \frac{1}{d_1^2\rho\kappa_1}\prod_{i\in[m_c]}\prod_{l\in L_i}\left(1+\epsilon_i(\mathbf{x}^l)\frac{c_{ij}}{\mathrm{rate}_j(\mathbf{x}^l)}\right) \;\ge\; \frac{1}{d_1^2\rho\kappa_1}\prod_{i\in[m_c]}\prod_{l\in L_i}\exp\left(\epsilon_i(\mathbf{x}^l)\frac{c_{ij}}{e\,\mathrm{rate}_j(\mathbf{x}^l)}\right).$$

where the last inequality is since $\epsilon_i c_{ij}/\mathrm{rate}_j \le 1/(3\ln(em)) \le 1$ and for $0\le a\le 1$, $e^{a/e}\le 1+a$. Multiplying on both sides by $d_1^2\rho\kappa_1$, taking the natural log, and reversing the inequality,

$$\sum_{i\in[m_c]}\sum_{l\in L_i}\epsilon_i(\mathbf{x}^l)\frac{c_{ij}}{e\mathrm{rate}_j(\mathbf{x}^l)} \;\le\; \ln\left(\frac{\mu d_1^2\rho\kappa_1}{\min_{i:c_{ij}>0} c_{ij}}\right) \;\le\; \ln\left(\mu d_1^2\rho\kappa\right)$$

and multiplying both sides by $e\cdot\max_{l\in L}\mathrm{rate}_j(\mathbf{x}^l)$,

$$\sum_{i\in[m_c]}\sum_{l\in L_i}\epsilon_i(\mathbf{x}^l)c_{ij} \;\le\; e\max_{l\in L}\mathrm{rate}_j(\mathbf{x}^l)\ln\left(\mu d_1^2\rho\kappa\right)\,. \tag{13}$$

Thus from (12) and (13), $(\mathbf{C^T y})_j \le e^2\max_{l\in L}\mathrm{rate}_j(\mathbf{x}^l)\ln\left(\mu d_1^2\rho\kappa\right) \le \sigma\max_{l\in L}\mathrm{rate}_j(\mathbf{x}^l)$. We will now show that $\max_{l\in L}\mathrm{rate}_j(\mathbf{x}^l)\le(\tilde{\mathbf{P}}^{\mathbf{T}}\mathbf{z})_j$, completing the proof. This follows since

$$\max_{l\in L}\mathrm{rate}_j(\mathbf{x}^l) \;=\; \max_{l\in L}\frac{\sum_{k\in[m]}\tilde{p}_{kj}\exp((\tilde{\mathbf{P}}\mathbf{x}^l)_k)}{\exp(\mathrm{est}(\mathbf{x}^l))} \;\le\; \sum_{k\in[m]}\tilde{p}_{kj}\max_{l\in L}\frac{\exp((\tilde{\mathbf{P}}\mathbf{x}^l)_k)}{\exp(\mathrm{est}(\mathbf{x}^l))} \;=\; \sum_{k\in[m]}\tilde{p}_{kj}z_k\,.$$

□

We now use the previous lemmas to prove the bound on the competitive ratio of our algorithm.

Lemma 9. *If $\Gamma \geq 2\sigma OPT$, then* MPC-APPROX *does not fail.*

Proof. Let $\mathbf{x}^f$ and ($\mathbf{y}^f$, $\mathbf{z}^f$) be the values for the primal and dual variables when at line 10 in the algorithm. $\mathbf{x}^f$ may be infeasible for the primal since the current job may not yet be assigned, however, ($\mathbf{y}^f$, $\mathbf{z}^f$) are feasible for the dual. Let $\mathbf{x}^*$ and ($\mathbf{y}^*$, $\mathbf{z}^*$) be the optimal solution. Then

$$OPT \;=\; \lambda(\mathbf{x}^*) \;=\; \sum_{i\in[m_c]} y_i^* \tag{14}$$

where the last equality follows from LP strong duality. For convenience of notation, let $\nu := \ln(em) + \tilde{\lambda}(\mathbf{x}^f)$. Since $\mathbf{x}$ is non-decreasing, $\tilde{\lambda}(\mathbf{x}^f) = \max_l \tilde{\lambda}(\mathbf{x}^l)$. Then by Lemmas 7 and 8, $\mathbf{z}^f/\nu$ and $\mathbf{y}^f \cdot \Gamma/(\sigma\nu)$ are feasible values for the dual variables. Thus the optimal dual value $\sum_i y_i^*$ is at least as large as $\sum_i y_i^f \cdot \Gamma/(\sigma\nu)$. From (5), $OPT \geq \max_{k,j} p_{kj}/d_1\rho\kappa_1$. Hence if $\Gamma \geq OPT$, the condition for Corollary 6 is satisfied. From (14) and Corollary 6,

$$OPT \;\geq\; \sum_i y_i^f \cdot \Gamma/(\sigma\nu) \;\geq\; (\mathrm{est}(\mathbf{x}^f) - \ln m - \frac{OPT}{\Gamma}) \cdot \Gamma/(\sigma\nu)\,,$$

or, rearranging terms,

$$\frac{\sigma\nu}{\Gamma} OPT + \frac{OPT}{\Gamma} + \ln m \;\geq\; \mathrm{est}(\mathbf{x}^f)\,.$$

Substituting the value of ν, and since $\mathrm{est}(\mathbf{x}^f) \geq \tilde{\lambda}(\mathbf{x}^f)$,

$$\frac{OPT}{\Gamma}\sigma\left(\ln(em) + \tilde{\lambda}(\mathbf{x}^f)\right) + \frac{OPT}{\Gamma} + \ln m \;\geq\; \tilde{\lambda}(\mathbf{x}^f)\,. \tag{15}$$

Using the bound on OPT from the statement of the lemma, and since $\sigma \geq 1$,

$$\frac{\ln(em) + \tilde{\lambda}(\mathbf{x}^f)}{2} + \ln(em) \;>\; \tilde{\lambda}(\mathbf{x}^f)\,,$$

and simplifying yields $\tilde{\lambda}(\mathbf{x}^f) < 3\ln(em)$. Hence, if $\Gamma \geq 2\sigma OPT$, the algorithm does not fail. □

Lemma 10. *If $4\sigma OPT \geq \Gamma \geq \max_{k,j} p_{kj}/(d_1\rho\kappa_1)$ and* MPC-APPROX *does not fail, it returns a $8\sigma\ln(em)$-competitive solution.*

Proof. Since $\Gamma \geq \max_{k,j} p_{kj}/(d_1\rho\kappa_1)$, from (15),

$$OPT\sigma\left(\ln(em) + \tilde{\lambda}(\mathbf{x}^f)\right) + OPT + \Gamma\ln m \;\geq\; \Gamma\tilde{\lambda}(\mathbf{x}^f) \;=\; \lambda(\mathbf{x}^f)\,.$$

Using the upper bound on Γ, and since $\tilde{\lambda}(\mathbf{x}^f) \leq 3\ln(em)$ by Corollary 4,

$$4OPT\sigma\ln(em) + OPT + 4OPT\sigma\ln m \;\geq\; \lambda(\mathbf{x}^f)\,.$$

This proves the lemma. □

Since $OPT \geq \max_{k,j} p_{kj}/d_1\rho\kappa_1$, Lemmas 9 and 10 imply

Theorem 11. *If $4\sigma OPT \geq \Gamma \geq 2\sigma OPT$, then* MPC-APPROX *does not fail and returns a $8\sigma\ln(em)$-competitive solution.*

2.2 Proceeding Without an Estimate on OPT.

We now discuss how to proceed without an estimate on OPT. We use a doubling procedure commonly used in online algorithms. We initially set $\Gamma = \max_{k,j:p_{kj}>0} p_{kj}/(d_1\rho\kappa_1)$ and use this value to scale the packing constraints. We run Algorithm MPC-APPROX with the scaled values. If the algorithm fails, we double Γ, scale the packing constraints by the new value of Γ and run the algorithm again. We repeat this each time the algorithm fails.

Each execution of Algorithm MPC-APPROX is a *trial*. Each trial τ has distinct primal and dual variables $(\lambda(\tau), \mathbf{x}(\tau))$ and $(\mathbf{y}(\tau), \mathbf{z}(\tau))$ that are initialized at the start of the trial and increase as the trial proceeds. At the start of the trial, each $x_j(\tau)$ is initialized to $x_j^0(\tau) = 1/(d_1^2\rho\kappa_1)$. If a trial fails, we double the value of Γ and proceed with the next trial with new primal and dual variables. Thus in every trial, $\Gamma \geq \max_{k,j:p_{kj}>0} p_{kj}/(d_1\rho\kappa_1)$.

Our final value for $(\mathbf{x}, \lambda)$ is the sum of the values obtained in each trial. Thus, our variables are non-decreasing. Let $\Gamma(\tau)$ be the value of Γ used in trial τ, and $\lambda^f(\tau)$ be the value of the primal $\lambda(\tau)$ when trial τ ends. T is the last trial, i.e., the algorithm does not fail in trial T. Since $\mathbf{x}$ obtained by the algorithm is the sum of $\mathbf{x}(\tau)$ in each trial τ, the value of the primal objective obtained by the algorithm is at most $\sum_{\tau\leq T} \lambda^f(\tau)$. Then

Theorem 12. *The value of the primal objective $\sum_{\tau\leq T} \lambda^f(\tau)$ obtained is $O(\ln m \ln(d\rho\kappa))OPT$.*

We first show a bound on Γ in any trial.

Lemma 13. *In any trial, $\Gamma \leq 4\sigma OPT$.*

Proof. Initially, $\Gamma = \max_{k,j:p_{kj}>0} p_{kj}/(d_1\rho\kappa_1) \leq OPT$ by (5). Hence the lemma is true for the first trial. Since Γ is doubled after each failed trial, by Lemma 9 some trial with $\Gamma \leq 4\sigma OPT$ will not fail. Hence, for every trial, $\Gamma \leq 4\sigma OPT$. □

Proof of Theorem 12. Define $\tilde{\lambda}^f(\tau) := \lambda^f(\tau)/\Gamma(\tau)$. By Corollary 4, $\tilde{\lambda}(\tau) \leq 3\ln(em)$ at the start of any phase. Within a phase, each variable gets multiplied by at most a factor of $\mu = 1 + 1/(3\ln(em))$. Hence when trial τ fails, $\tilde{\lambda}^f(\tau) = \lambda^f(\tau)/\Gamma(\tau) \leq 1 + 3\ln(em) \leq 4\ln(em)$, or $\lambda^f(\tau) \leq 4\Gamma(\tau)\ln(em)$. Since the value of $\Gamma(\tau)$ doubles after each trial,

$$\sum_{\tau\leq T} \lambda^f(\tau) \;\leq\; 4\ln(em) \sum_{\tau\leq T} \Gamma(\tau) \;=\; 4\ln(em) \sum_{\tau\leq T} 2^{\tau-T}\Gamma(T) \;\leq\; 8\ln(em)\Gamma(T)\,. \tag{16}$$

Thus, from (16) and Lemma 13, $\sum_{\tau\leq T} \lambda^f(\tau) \leq 32\sigma\ln(em)OPT$, proving the theorem. □

2.3 A Lower Bound for Mixed Packing and Covering Online

We give a lower bound on the competitive ratio of any deterministic algorithm for online mixed packing and covering. Given upper bounds m and d on the number of packing constraints and on the number of variables in any (packing or covering) constraint respectively, we give an example to show the following lower bound.

Theorem 14. *Any deterministic algorithm for OMPC is $\Omega(\log(d/\log m)\log m)$-competitive.*

Our algorithm for OMPC in Section 2.1 is thus nearly tight. For parameters d and m, we give an example which has m packing constraints, at most $2d$ variables in each covering constraint, and at most $d\log m$ variables in each packing constraint. For this example, we show that $OPT = 1$ and any deterministic algorithm gets value $\Omega(\log d \log m)$. The theorem follows.

We assume that both d and m are powers of 2 without loss of generality, otherwise we redefine d to be the highest power of 2 that is at most the given value of d, and redefine m similarly. Our example has $2(m-1)d$ variables. We partition the variables into $2(m-1)$ pairwise disjoint sets, with each set consisting of d variables, and use B_i to refer to the ith set. We refer to these sets as *blocks*. For any set of variables S, we use $w(S)$ to refer to the sum of the values assigned by the algorithm to the variables in S, and use $\Sigma(S)$ to refer to the expression $\sum_{x \in S} x$.

We first show how given two blocks B_1 and B_2 of size d, we can construct covering constraints so that $w(B_i) \geq H_d/2$ for one of $i \in \{1, 2\}$, while the constraints can be satisfied by setting $x_j = 1$ for a single variable $x_j \in B_{i'}$, $i' \neq i$. The covering constraints are given by Algorithm 2. H_d refers to the dth harmonic number.

Algorithm 2 Given blocks B_1 and B_2:

1: $B'_1 \leftarrow B_1$, $B'_2 \leftarrow B_2$
2: **for** $i = 1 \rightarrow d - 1$ **do**
3: Offer the covering constraint $\Sigma(B'_1 \cup B'_2) \geq 1$
4: Let x_1, x_2 be the variables assigned maximum value in B'_1 and B'_2 respectively
5: $B'_1 \leftarrow B'_1 \setminus \{x_1\}$, $B'_2 \leftarrow B'_2 \setminus \{x_2\}$
6: Offer the covering constraint $\Sigma(B'_1 \cup B'_2) \geq 1$

Lemma 15. *Either* $w(B_1) \geq H_d/2$ *or* $w(B_2) \geq H_d/2$.

Proof. Let x_1, x_2 be the variables assigned maximum value in B'_1 and B'_2 respectively in the ith iteration of the for loop. Since $|B'_1| = |B'_2| = (d + 1 - i)$, and x_1, x_2 have maximum value in B'_1 and B'_2 respectively, $x_1 \geq w(B'_1)/(d + 1 - i)$ and $x_2 \geq w(B'_2)/(d + 1 - i)$. Further, since $w(B'_1) + w(B'_2) \geq 1$, $x_1 + x_2 \geq 1/(d + 1 - i)$. Thus when all the covering constraints are satisfied, $w(B_1 \cup B_2) \geq H_d$, and hence either $w(B_1) \geq H_d/2$ or $w(B_2) \geq H_d/2$. □

Assume $w(B_1) \geq H_d/2$. Then there exists some variables $x_j \in B_2$ which is in each covering constraint introduced, and hence all the constraints can be satisfied by setting this variable to 1.

For the complete example, consider a complete binary tree with m leaf nodes. Each node in this tree except the root corresponds to a block, and no two nodes correspond to the same block. Our packing constraints correspond to the leaf nodes, with packing constraint k being $E(\cup_{i \in Q_k} B_i) \leq \lambda$ where Q_k is the set of blocks encountered on the path from the root to the leaf node corresponding to packing constraint k.

For a node v, let l and r be the left and right child respectively, and let B_l and B_r be the blocks corresponding to these children. We now start from the root node and walk to a leaf node in the following way. When we are at node v, we run Algorithm 2 with blocks B_l and B_r. If $w(B_l) \geq w(B_r)$ we step on the left child and "mark" the right child, else we step on the right child and "mark" the left child. We continue with the node we stepped on as node v, and continue in this manner until we reach a leaf node. Then say the leaf node we arrive at corresponds to packing constraint k. Since each block on the path from the root to this leaf node (except the root) has weight at least $H_d/2$ by Lemma 15, and the path from the root to any leaf has $\log n$ nodes, the total value assigned by the algorithm to variables in this constraint is at least $\log m \cdot H_d/2$, thus $\lambda \geq \log m \cdot H_d/2$.

On the other hand, setting a single variable to 1 in each marked node satisfies all the covering constraints. The path from the root to any leaf node contains at most one marked node, since when we mark a node, the blocks in the subtree rooted at that node do not appear in any covering constraint. Thus, we can satisfy the covering constraints by setting at most a single variable to 1 in each packing constraint, where for a packing constraint, the variable set is in the marked node (if any) in the path from the root to

the leaf corresponding to the packing constraint. Hence OPT = 1, and any deterministic algorithm obtains $\lambda \geq \log m \cdot H_d/2$.

3 UMSC and CCFL

We now build upon the techniques in Section 2 and give a polylogarithmic-competitive integral algorithm for UMSC and CCFL. Recall that both UMSC and CCFL generalize online set-cover, for which there is a lower bound of $\Omega(\log m \log n)$ on the competitive ratio assuming BPP $\neq$ NP [3]. Our algorithm is thus tight modulo a logarithmic factor.

CCFL generalizes UMSC; an instance of UMSC is an instance of CCFL where each facility corresponds to a machine and has unit capacity, each client corresponds to a job, and all assignment costs are zero. We describe an algorithm for CCFL, which also gives an algorithm for UMSC with the same competitive ratio. Further, in CCFL, the demand p_{ij} and capacity u_i only appear as the ratio p_{ij}/u_i. We redefine p_{ij} as this ratio, and assume that the capacity of every facility is unity. The congestion of a facility is then the sum of the demands of clients assigned to the facility.

We use $[m]$ to denote the set of facilities, and $[n]$ to denote the set of clients. We exclude trivial instances and assume m, n are at least 2. We assume that n is given offline. Variables i, i' index facilities, while j, j' index clients. Clients appear in order of their indices: the first client is client 1, and the last client is client n. The *total cost* Z of an assignment of clients to facilities is the sum of the maximum congestion, fixed-charge, and assignment costs. Z^* is the total cost of the optimal assignment. We will assume we are given an estimate Z with $Z^* \leq Z \leq 2Z^*$. In the absence of this estimate, we use a doubling approach as described previously; Section 3.4 explains how doubling can be used for this particular problem. For client j, $F_j(Z) := \{i : p_{ij} + a_{ij} + c_i \leq Z\}$. Since assigning client j to a facility i not in $F_j(Z)$ would incur total cost larger than Z, we fix the fractional assigment of client j to any facility i not in $F_j(Z)$ to be zero.

We first give an algorithm that obtains a fractional solution for the problem, and then use a randomized rounding technique adapted from [26] to obtain an integral solution. A fractional solution corresponds to a solution to linear program CCFL-LP1(Z), which takes Z as a parameter. A client may be fractionally assigned to facilities, and the sum of these fractional assignments for each client must be at least 1. x_{ij} is the fractional assignment of client j to facility i. Facilities can also be opened fractionally, and y_i is the fraction to which facility i is open. The fraction to which any facility is opened is an upper bound on both the fractional assignment of any client to that facility, and the ratio of congestion of the facility to Z. λ is an upper bound on y_i for each facility. Since for every facility the fraction y_i is an upper bound on the ratio of congestion to Z, $Z\lambda$ is the maximum congestion of any facility.

$$\textbf{CCFL-LP1}(Z)\textbf{:} \quad \min \sum_{i\in[m]} c_i y_i + Z\lambda + \sum_{i\in[m],\, j\in[n]} a_{ij} x_{ij}$$

$$\begin{aligned}
\textstyle\sum_{i\in[m]} x_{ij} &\geq 1, && \forall j \in [n] \\
y_i - x_{ij} &\geq 0, && \forall i \in [m], j \in [n] \\
Z y_i - \textstyle\sum_{j\in[n]} p_{ij} x_{ij} &\geq 0, && \forall i \in [m] \\
\lambda - y_i &\geq 0, && \forall i \in [m] \\
\lambda &\geq 1
\end{aligned}$$

$\mathbf{x}$ is the vector $(x_{ij})_{i\in[m],\, j\in[n]}$, and similarly $\mathbf{y}$ is the vector $(y_i)_{i\in[m]}$. $\mathcal{OPT}_1(Z)$ is the cost of the optimal solution to CCFL-LP1(Z). Since we restrict assignment of client j to facilities in $F_j(Z)$, if $F_j(Z) = \emptyset$, CCFL-LP1(Z) is infeasible. We assume that $Z^* \leq Z \leq 2Z^*$, and hence CCFL-LP1(Z) is feasible. Also, since $\lambda \geq 1$,

Fact 16. $\mathcal{OPT}_1(Z) \geq Z$.

Define $\rho := \max_j \frac{\max_i(c_i+p_{ij}+a_{ij})}{\min_i(c_i+p_{ij}+a_{ij})}$. We use techniques from Section 2 and obtain an $O(\log(mn)\log(mn\rho))$-competitive fractional solution for CCFL-LP1(Z). We begin by highlighting the major differences between CCFL and OMPC, and briefly mention how they are dealt with.

Firstly, the linear programm CCFL-LP1(Z) no longer consists solely of packing and covering constraints, since there are variables with negative coefficients. However, we can still express the objective of CCFL-LP1(Z) as a function of the vector $\mathbf{x}$: given $\mathbf{x}$ that satisfies the first set of constraints in CCFL-LP1(Z), define $y_i(\mathbf{x})$ as the minimum value of y_i that satisfies the remaining constraints; $\lambda(\mathbf{x})$ is defined correspondingly. We then proceed as in MPC-APPROX: we define $\mathrm{cost}(\mathbf{x})$ as a derivable approximation to the objective, and $\mathrm{rate}(\mathbf{x})$ as the derivative of $\mathrm{cost}(\mathbf{x})$. $\mathrm{rate}(\mathbf{x})$ is then used to obtain the multiplicative updates for variables $\mathbf{x}$.

Secondly, whereas earlier $\mathrm{rate}(\mathbf{x})$ was a continuous function of $\mathbf{x}$ for OMPC, this is no longer the case for CCFL. Now $\mathrm{rate}(\mathbf{x})$ depends on whether the second set of constraints in CCFL-LP1(Z) are tight. We deal with this by separating the updates where the second set of constraints is tight, and increment $\mathbf{x}$ differently in each case.

Thirdly, in Section 2 for each variable x_j since the packing constraints are available offline, the coefficients of x_j are also known offline. This allows us to initialize variables offline. In CCFL, clients arrive online, and when each client arrives, we learn its demand and assignment cost for each facility. Thus the coefficients d_{ij} and c_{ij} of each variable x_{ij} in CCFL-LP1(Z) are received online, and these variables need to be initialized online. We show that the increase in $\mathrm{cost}(\mathbf{x})$ because of these initializations is small.

Fourthly, CCFL-LP1(Z) is a parametric LP. If we do not know Z^*, we use a doubling procedure twice: once to obtain Z such that $Z^* \leq Z \leq 2Z^*$, and once again to scale CCFL-LP1(Z) by Γ, as in Section 2.

3.1 A Fractional Solution for CCFL

We start by scaling the CCFL-LP1(Z) by a parameter Γ to obtain LP2(Z, Γ) and its dual, D2(Z, Γ). $\mathcal{OPT}_2(Z, \Gamma)$ is the cost of the optimal solution to LP2(Z, $\tilde{\lambda}$). In the following analysis we will keep the dual variable $\mu = 0$ and exclude it from further discussion.

LP2(Z, Γ):
$$\begin{aligned} \min \ & \sum_i c_i\tilde{y}_i + Z\tilde{\lambda} + \frac{1}{\Gamma}\sum_{i,j} a_{ij}x_{ij} \\ \sum_i x_{ij} &\geq 1, \ \forall j \\ \tilde{y}_i - \frac{x_{ij}}{\Gamma} &\geq 0, \ \forall i,j \\ Z\tilde{y}_i - \sum_j p_{ij}\frac{x_{ij}}{\Gamma} &\geq 0, \ \forall i \\ \tilde{\lambda} - \tilde{y}_i &\geq 0, \ \forall i \\ \tilde{\lambda} &\geq 1 \end{aligned}$$

D2(Z, Γ):
$$\begin{aligned} \max \ & \sum_j \alpha_j + \frac{1}{\Gamma}\mu \\ \Gamma\alpha_j - \beta_{ij} - p_{ij}\gamma_i - a_{ij} &\leq 0, \ \forall i,j \\ \sum_j \beta_{ij} + Z\gamma_i - \delta_i &\leq c_i, \ \forall i \\ \sum_i \delta_i + \mu &\leq Z \end{aligned}$$

Fact 17. *For any Z, $\Gamma > 0$, $(\mathbf{x}, \mathbf{y}, \lambda)$ is a feasible solution to CCFL-LP1(Z) of cost Z' iff $(\mathbf{x}, \mathbf{y}/\Gamma, \lambda/\Gamma)$ is a feasible solution to LP2(Z, Γ) of cost Z'/Γ.*

Given a vector $\mathbf{x}$, let $\tilde{w}_i(\mathbf{x}) := \max_j x_{ij}/\Gamma$, $\tilde{v}_i(\mathbf{x}) := \sum_j p_{ij}x_{ij}/(Z\Gamma)$. Define $\tilde{y}_i(\mathbf{x}) := \tilde{v}_i(\mathbf{x}) + \tilde{w}_i(\mathbf{x})$, and $\tilde{\lambda}(\mathbf{x}) := \max_i \tilde{y}_i(\mathbf{x})$. If $\mathbf{x}$ satisfies $\sum_i x_{ij} \geq 1$ for all j, then $(\mathbf{x}, \tilde{\mathbf{y}}(\mathbf{x}), \tilde{\lambda}(\mathbf{x}))$ is feasible for LP2(Z, Γ), where $\tilde{\mathbf{y}}$ is the m-vector of values $(\tilde{y}_i(\mathbf{x}))_{i\in[m]}$. As in Section 2,

$$\text{est}(\mathbf{x}) \;:=\; \ln \sum_i \left(e^{\sum_j \frac{p_{ij}x_{ij}}{Z\Gamma}} \right) + \ln \left(\sum_{i,j} e^{\frac{x_{ij}}{\Gamma}} \right) \tag{17}$$

is an upper bound on $\tilde{\lambda}(\mathbf{x})$, and $\text{cost}(\mathbf{x}) := Z \cdot \text{est}(\mathbf{x}) + \sum_i c_i \tilde{y}_i(\mathbf{x}) + \sum_{i,j} a_{ij}x_{ij}/\Gamma$ is an upper bound on the cost of the solution $(\mathbf{x}, \tilde{\mathbf{y}}(\mathbf{x}), \tilde{\lambda}(\mathbf{x}))$. The rate of change of $\text{cost}(\mathbf{x})$ is

$$\text{rate}_{ij}(\mathbf{x}) \;:=\; \frac{\partial \text{cost}(\mathbf{x})}{\partial x_{ij}} \;=\; Z\frac{\partial \text{est}(\mathbf{x})}{\partial x_{ij}} + \frac{c_i}{\Gamma}\left(\frac{p_{ij}}{Z} + 1_{ij}\right) + \frac{a_{ij}}{\Gamma} \tag{18}$$

where $1_{ij} = 1$ if $x_{ij} = \Gamma \tilde{w}_i$, and 0 otherwise.

Our algorithm is given as ASSIGN(Γ) . Define

$$x^0_{ij} \;:=\; \frac{1}{2mn} \frac{\min_{i'}(c_{i'} + p_{i'j} + a_{i'j})}{c_i + p_{ij} + a_{ij}}\,. \tag{19}$$

At the beginning of the algorithm, before any requests arrive, set $x_{ij} = 0$ for all i, j. When client j arrives, we initialize $x_{ij} = x^0_{ij}$ for all i. As long as j is not fully assigned, we increment the fractional assignment x_{ij} for each client i. The increment occurs in *phases* where a phase is a single iteration of the while loop in the algorithm. The phases are indexed by l. L is the set of all phase indices, and L_j is the set of phase indices for phases executed to assign client j.

The increase in x_{ij} in each phase is inversely proportional to $\text{rate}_{ij}(\mathbf{x})$. Define $\mu := 1 + \frac{1}{6\ln(emn)}$. We also scale each update for client j by $\epsilon_j(\mathbf{x})$, where

$$\epsilon_j(\mathbf{x}) := (\mu - 1) \min_i \text{rate}_{ij}(\mathbf{x})\,. \tag{20}$$

This definition ensures that in each phase, the factor by which each variable is incremented is at most μ.

Algorithm 3 ASSIGN(Γ) : When client j arrives

1: $\forall i$, $x_{ij} \leftarrow x^0_{ij}$
2: **while** $\sum_i x_{ij} < 1$ **do**
3: $\quad l \leftarrow l + 1$, $\mathbf{x}^l \leftarrow \mathbf{x}$
4: $\quad$ **for** $i \in F_j(Z)$ **do**
5: $\quad\quad$ **if** $\tilde{w}_i(\mathbf{x}^l) > x_{ij}/\Gamma$ **then**
6: $\quad\quad\quad x_{ij} \leftarrow \min\left\{\Gamma w_i(\mathbf{x}^l), x_{ij}\left(1 + \frac{\epsilon_j(\mathbf{x}^l)}{\text{rate}_{ij}(\mathbf{x}^l)}\right)\right\}$ /* see definitions in (18), (20) */
7: $\quad\quad$ **else**
8: $\quad\quad\quad x_{ij} \leftarrow x_{ij}\left(1 + \frac{\epsilon_j(\mathbf{x}^l)}{\text{rate}_{ij}(\mathbf{x}^l)}\right)$
9: $\quad \alpha_j \leftarrow \alpha_j + e\epsilon_j(\mathbf{x}^l)$ /* for analysis */
10: $\quad$ **if** $\text{cost}(\mathbf{x}) > 5Z\ln(emn)$ **then return** FAIL

Fact 18. *At any stage in the algorithm, $x_{ij} \le \mu$ for all i, j.*

Our analysis of ASSIGN(Γ) follows the primal-dual analysis in Section 2 closely. One difference between the current problem and Section 2 is that since we receive requests online, we do not know the coefficients of variables in the packing constraints offline, and unlike Section 2 cannot initialize our variables offline. In ASSIGN(Γ) as each request is received, we obtain the corresponding coefficients, and initialize our variables $x_{ij} = x^0_{ij}$ in line 1. For client j, define init_j as the change in $\text{cost}(\mathbf{x})$ on execution of line 1 when j arrives. $\text{cost}(\mathbf{x})$ is initially $\text{cost}(\mathbf{0})$ and increases either due to line 1 or within a phase. We begin our analysis by showing bounds on the change in $\text{cost}(\mathbf{x})$ due to these.

Lemma 19. *If $\Gamma \geq 1$, then $\text{init}_j \leq Z/n$ for any client j.*

Proof. Fix a client j. Let $\mathbf{x}'$ and $\mathbf{x}''$ be the values of $\mathbf{x}$ before and after the execution of line 1 on arrival of client j. We consider the differences $Z(\text{est}(\mathbf{x}'') - \text{est}(\mathbf{x}'))$ and $\sum_i c_i(\tilde{v}_i(\mathbf{x}'') + \tilde{w}_i(\mathbf{x}'')) - \sum_i c_i(\tilde{v}_i(\mathbf{x}') + \tilde{w}_i(\mathbf{x}')) + \sum_{i,j'} a_{ij'}(x''_{ij'} - x'_{ij'})/\Gamma$ separately. Since $\text{cost}(\mathbf{x}) = \sum_i c_i(\tilde{v}_i(\mathbf{x}) + \tilde{w}_i(\mathbf{x})) + Z\text{est}(\mathbf{x}) + \frac{1}{\Gamma}\sum_{i,j} a_{ij}x_{ij}$, the sum of these differences will give us init_j.

By definition of $\text{est}(\mathbf{x})$,

$$\begin{aligned}\text{est}(\mathbf{x}'') - \text{est}(\mathbf{x}') \;=\; & \ln \sum_i \exp\left(\frac{\sum_{j'} p_{ij'} x''_{ij'}}{Z\Gamma}\right) - \ln \sum_i \exp\left(\frac{\sum_{j'} p_{ij'} x'_{ij'}}{Z\Gamma}\right) \\ & + \ln \sum_{i,j'} \exp\left(\frac{x''_{ij'}}{\Gamma}\right) - \ln \sum_{i,j'} \exp\left(\frac{x'_{ij'}}{\Gamma}\right).\end{aligned}$$

$\mathbf{x}''$ and $\mathbf{x}'$ differ only in values for client j, and are identical for other requests $j' \neq j$. Further, for all i and $j' > j$, $x''_{ij'} = x'_{ij'} = 0$. Thus

$$\begin{aligned}\text{est}(\mathbf{x}'') - \text{est}(\mathbf{x}') \;=\; & \ln \sum_i \exp\left(\frac{\sum_{j'<j} p_{ij'} x'_{ij'} + p_{ij} x^0_{ij}}{Z\Gamma}\right) - \ln \sum_i \exp\left(\frac{\sum_{j'<j} p_{ij'} x'_{ij'}}{Z\Gamma}\right) \\ & + \ln \sum_i \left(\sum_{j'<j} \exp \frac{x'_{ij'}}{\Gamma} + \exp \frac{x^0_{ij}}{\Gamma} + (n-j-1)\right) - \ln \sum_i \left(\sum_{j'<j} \exp \frac{x'_{ij'}}{\Gamma} + (n-j)\right).\end{aligned}$$

Since $\Gamma \geq 1$ by assumption, $x^0_{ij}/\Gamma \leq 1/(2mn)$. Further, since $p_{ij} \leq Z$ for $i \in F_j(Z)$, $p_{ij}x^0_{ij}/(Z\Gamma) \leq x^0_{ij}/\Gamma \leq 1/(2mn)$. Substituting in the previous expression for $\text{est}(\mathbf{x}'') - \text{est}(\mathbf{x}')$ yields

$$\begin{aligned}\text{est}(\mathbf{x}'') - \text{est}(\mathbf{x}') \;\leq\; & \ln\left(\exp\frac{1}{2mn} \sum_i \exp \frac{\sum_{j'<j} p_{ij'} x'_{ij'}}{Z\Gamma}\right) - \ln \sum_i \exp\left(\frac{\sum_{j'<j} p_{ij'} x'_{ij'}}{Z\Gamma}\right) \\ & + \ln \frac{\sum_i \left(\sum_{j'<j} \exp \frac{x'_{ij'}}{\Gamma} + \exp\frac{1}{2mn} + (n-j-1)\right)}{\sum_i \left(\sum_{j'<j} \exp \frac{x'_{ij'}}{\Gamma} + (n-j)\right)} \\ =\; & \frac{1}{2mn} + \ln\left(1 + \frac{\sum_i (\exp\frac{1}{2mn} - 1)}{\sum_i \left(\sum_{j'<j} \exp \frac{x'_{ij'}}{\Gamma} + (n-j)\right)}\right) \\ \leq\; & \frac{1}{2mn} + \ln\left(1 + \frac{\sum_i (\exp\frac{1}{2mn} - 1)}{\sum_i n}\right) \;\leq\; \frac{1}{2mn} + \frac{\exp\frac{1}{2mn} - 1}{n}\end{aligned}$$

where the last inequality is because $1 + a \leq \exp(a)$ for all $a \in \mathbb{R}$. Using $m, n \geq 2$,

$$Z(\text{est}(\mathbf{x}'') - \text{est}(\mathbf{x}')) \;\leq\; \frac{Z}{4n} + \frac{Z\exp(\frac{1}{8}) - 1}{n} \;\leq\; \frac{Z}{2n}. \tag{21}$$

For the second bound, since $x''_{ik} = x'_{ik}$ for $k \neq j$, $\tilde{w}_i(\mathbf{x}'') - \tilde{w}_i(\mathbf{x}') \leq x''_{ij}/\Gamma = x^0_{ij}/\Gamma$. Further since $p_{ij} \leq Z$ for $i \in F_j(Z)$, $\tilde{v}_i(\mathbf{x}'') - \tilde{v}_i(\mathbf{x}') = \frac{p_{ij}x^0_{ij}}{Z\Gamma} \leq x^0_{ij}/\Gamma$. Hence

$$\begin{aligned}\sum_i \left(c_i(\tilde{v}_i(\mathbf{x}'') + \tilde{w}_i(\mathbf{x}'') - \tilde{v}_i(\mathbf{x}') - \tilde{w}_i(\mathbf{x}')) + \sum_{j'} \frac{a_{ij'}}{\Gamma}(x''_{ij'} - x'_{ij'}) \right) &\leq \sum_i \frac{x^0_{ij}}{\Gamma}(2c_i + a_{ij}) \\ &\leq 2Z\frac{x^0_{ij}}{\Gamma} \\ &\leq \frac{Z}{2n} \end{aligned} \tag{22}$$

where the second inequality is because $c_i + a_{ij} \leq Z$ for $i \in F_j(Z)$, and the last inequality is by definition of x^0_{ij} and since $\Gamma \geq 1$ and $m \geq 2$. From (21) and (22), the lemma follows. □

Corollary 20. *If $\Gamma \geq 1$, then at the beginning of every phase, $\mathrm{cost}(\mathbf{x}) \leq 6Z\ln(emn)$.*

Proof. When the previous phase ended, since ASSIGN(Γ) did not fail, $\mathrm{cost}(\mathbf{x}) \leq 5Z\ln(emn)$. Between phases, $\mathbf{x}$ is incremented by at most the arrival of a client, which increases $\mathrm{cost}(\mathbf{x})$ by at most Z/n by Lemma 19. □

We now show that the increase in $\mathrm{cost}(\mathbf{x})$ in any phase of the algorithm is bounded from above by the change in the dual objective.

Lemma 21. *If $\Gamma \geq 1$, the increase in $\sum_j \alpha_j$ is an upper bound on the increase in $\mathrm{cost}(\mathbf{x})$ in every phase.*

Proof. We show the lemma for phase l, corresponding to request r. Let cost^l and cost^{l+1} be the values of $\mathrm{cost}(\mathbf{x})$ before and after the variables are incremented in phase l respectively. We will show that $\mathrm{cost}^{l+1} - \mathrm{cost}^l \leq e\,\epsilon_j(\mathbf{x}^l)$. Since $e\,\epsilon_j(\mathbf{x}^l)$ is the increase in the dual objective, this will prove the lemma.

Our proof follows the proof for Lemma 5. Let $\mathbf{x}^l$ and $\mathbf{x}^{l+1}$ be the values of $\mathbf{x}$ before and after being incremented in phase l. In phase l, only the variables correpsonding to client j get incremented. We ignore variables for the other requests, and for each x_{ij}, let $g_i(t) := x^l_{ij} + (x^{l+1}_{ij} - x^l_{ij})t$ be defined for $0 \leq t \leq 1$. Note that $g_i(0) = x^l_{ij}$ and $g_i(1) = x^{l+1}_{ij}$. Define $\mathbf{g}(t) = (g_i(t))_{i \in F_j(Z)}$. With some abuse of notation, any function of $\mathbf{x}$, say $h(\mathbf{x})$, can be written as a function of t, with $h(t) := h(\mathbf{g}(t))$. Thus, the functions $\mathrm{cost}(\mathbf{x})$, $\mathrm{rate}_{ij}(\mathbf{x})$, $\tilde{y}_i(\mathbf{x})$ and $\tilde{\lambda}(\mathbf{x})$ can be written as functions of t. In particular, $\mathrm{rate}_{ij}(t) := d\,\mathrm{cost}(t)/dg_i(t)$.

We use these alternate expressions in the remainder of the proof. By the chain rule,

$$\frac{d\,\mathrm{cost}(t)}{dt} = \sum_i \frac{\partial \mathrm{cost}(t)}{\partial g_i(t)} \frac{dg_i(t)}{dt} = \sum_i \mathrm{rate}_{ij}(t) \frac{dg_i(t)}{dt}\,,$$

and hence,

$$\mathrm{cost}^{l+1} - \mathrm{cost}^l = \int_{t=0}^1 \frac{d\mathrm{cost}(t)}{dt} dt = \sum_i \int_{t=0}^1 \mathrm{rate}_{ij}(t) \frac{dg_i(t)}{dt} dt\,. \tag{23}$$

The function $\mathrm{rate}_{ij}(t)$ may not be continuous if for some i, $\Gamma\tilde{w}_i(\mathbf{x}^{l+1}) = x^{l+1}_{ij}$, but $\Gamma\tilde{w}_i(\mathbf{x}^l) > x^l_{ij}$. By our choice of updates in ASSIGN(Γ) , the discontinuity is only at the point $t = 1$; hence we redefine $\mathrm{rate}_{ij}(1) := \lim_{t\to 1^-} \mathrm{rate}_{ij}(t)$. Since we change $\mathrm{rate}_{ij}(t)$ at a single point, (23) is still true. By Corollary 20, $\mathrm{cost}^l \leq 6Z\ln(emn)$. Also, each variable gets incremented by at most a factor of μ in a phase. Then by Lemma 46, $\mathrm{rate}_{ij}(t) \leq e\,\mathrm{rate}_{ij}(0)$ for $0 \leq t \leq 1$, hence

$$\text{cost}^{l+1} - \text{cost}^l \;\le\; e\sum_i \text{rate}_{ij}(\mathbf{x}^l)\int_{t=0}^{1}\frac{dg_i(t)}{dt}dt \;=\; e\sum_i \text{rate}_{ij}(\mathbf{x}^l)(x_{ij}^{l+1} - x_{ij}^l)\,.$$

Since in phase l each variable is multiplied by $1 + \frac{\epsilon_j(\mathbf{x}^l)}{\text{rate}_{ij}(\mathbf{x}^l)}$,

$$\text{cost}^{l+1} - \text{cost}^l \;\le\; e\sum_i \text{rate}_{ij}(\mathbf{x}^l)\frac{\epsilon_j(\mathbf{x}^l)x_{ij}^l}{\text{rate}_{ij}(\mathbf{x}^l)} \;=\; e\sum_i \epsilon_j(\mathbf{x}^l)x_{ij}^l \;\le\; e\epsilon_j(\mathbf{x}^l)$$

where the last inequality follows since, on entering the for loop, $\sum_i x_{ij}^l < 1$. Since $e\epsilon_j(\mathbf{x}^l)$ is the increase in the dual objective, this proves the lemma. □

We now discuss our dual variables and show feasibility. By definition,

$$\frac{\partial \text{est}(\mathbf{x})}{\partial x_{ij}} \;=\; \frac{1}{\Gamma}\left[\frac{p_{ij}}{Z}\frac{\exp(\sum_{j'} p_{ij'}x_{ij'}/(Z\Gamma))}{\sum_{i'}\exp(\sum_{j'} p_{i'j'}x_{i'j'}/(Z\Gamma))} + \frac{\exp(x_{ij}/\Gamma)}{\sum_{i',j'}\exp(x_{i'j'}/\Gamma)}\right]. \tag{24}$$

We use the following notation. Recall that for a client j, $F_j(Z)$ is the set of facilities i with $c_i + p_{ij} + a_{ij} \le Z$, and $x_{ij} = 0$ for any $i \notin F_j(Z)$. For all j and $i \notin F_j(Z)$, we leave the corresponding dual variables undefined. For facility i, n_i is the index of the first client j so that $i \in F_j(Z)$.

$$\forall i,\; z_i(n_i - 1) \;:=\; x^0_{in_i}\,,\;\; \text{and } \forall j \ge n_i,\; z_i(j) \;:=\; \max_{j' \le j} x_{ij'}\,, \tag{25}$$

$$\begin{aligned}
\chi_{ij} \quad &:= \quad \max_{l \in L_j}\frac{\exp(x_{ij}^l/\Gamma)}{\sum_{i',j'}\exp(x_{i'j'}^l/\Gamma)} && \forall i \in [m],\; \forall j : i \in F_j(Z)\,, \\
\eta_i \quad &:= \quad \max_{l \in L}\frac{\exp(\sum_{j'} p_{ij'}x_{ij'}/(Z\Gamma))}{\sum_{i'}\exp(\sum_{j'} p_{i'j'}x_{i'j'}/(Z\Gamma))} && \forall i \in [m]\,.
\end{aligned} \tag{26}$$

From (24) and these definitions, for any i, j and any phase $l \in L_j$,

$$\frac{\partial \text{est}(\mathbf{x}^l)}{\partial x_{ij}} \;\le\; \frac{1}{\Gamma}\left(\frac{p_{ij}}{Z}\eta_i + \chi_{ij}\right). \tag{27}$$

Define $\sigma := 4e^2\ln(2\mu mn\rho)$. Set the dual variables:

$$\begin{aligned}
\beta_{ij} \;&=\; Z\chi_{ij}\frac{\sigma}{2e^2} + c_i \ln\frac{z_i(j)}{z_i(j-1)}\,, && \forall i \in [m],\; \forall j : i \in F_j(Z)\,, \\
\gamma_i \;&=\; \left(\eta_i + \frac{c_i}{Z}\right)\frac{\sigma}{2e^2}\,, && \forall i \in [m]\,, \\
\delta_i \;&=\; Z\left(\sum_{j : i \in F_j(Z)} \chi_{ij} + \eta_i\right)\frac{\sigma}{2e^2}\,, && \forall i \in [m]\,.
\end{aligned} \tag{28}$$

We define $z_i(j)$ for $j \in \{n_i - 1, ..., n\}$ since the definition of β_{in_i} requires $z_i(n_i - 1)$. We now show bounds on the infeasibility of each dual constraint in D2(Z, Γ) in the following sequence of lemmas.

Lemma 22. *For all i, j, $\alpha_j \le \frac{e^2}{\Gamma}\left(\beta_{ij} + p_{ij}\gamma_i + a_{ij}\frac{\sigma}{2e^2}\right)$.*

Proof. For any i, j, let $L_{ij}^{>}$ be the set of phases where $\tilde{w}_i > x_{ij}/\Gamma$ before x_{ij} is incremented, and $L_{ij}^{=}$ is the set of phases where $\tilde{w}_i = x_{ij}/\Gamma$ before x_{ij} is incremented. Then $L_{ij}^{>} \cup L_{ij}^{=} = L_j$. Let l be the first phase executed when $x_{ij}/\Gamma = \tilde{w}_i$ at the beginning of the phase. Then in every subsequent phase in L_j, $x_{ij}/\Gamma = \tilde{w}_i$ at the beginning of the phase. Hence any phase $l \in L_{ij}^{=}$ occurs after all the phases in $L_{ij}^{>}$.

In any phase in $L_{ij}^{>}$ except the last, x_{ij} is incremented by $(1+\epsilon_j(\mathbf{x}^l)/\mathrm{rate}_{ij}(\mathbf{x}^l))$. Before the last phase, $x_{ij} \leq 1$. In the last phase, x_{ij} is incremented by at most $(1 + \epsilon_j(\mathbf{x}^l)/\mathrm{rate}_{ij}(\mathbf{x}^l)) \leq \mu$. Hence

$$x_{ij}^0 \prod_{l \in L_{ij}^{>}} \left(1 + \frac{\epsilon_j(\mathbf{x}^l)}{\mathrm{rate}_{ij}(\mathbf{x}^l)}\right) \;\leq\; \mu\,.$$

Using $1 + a \geq e^{a/e}$ for $0 \leq a \leq 1$, and by rearranging the terms,

$$\exp\left(\sum_{l \in L_{ij}^{>}} \frac{\epsilon_j(\mathbf{x}^l)}{e\,\mathrm{rate}_{ij}(\mathbf{x}^l)}\right) \;\leq\; \frac{\mu}{x_{ij}^0}\,.$$

Taking the natural log on both sides, and observing that $x_{ij}^0 \geq 1/(2mn\rho)$,

$$\left(\sum_{l \in L_{ij}^{>}} \frac{\epsilon_j(\mathbf{x}^l)}{e\,\mathrm{rate}_{ij}(\mathbf{x}^l)}\right) \;\leq\; \ln\frac{\mu}{x_{ij}^0} \;\leq\; \ln(2\mu mn\rho)\,,$$

and hence

$$\sum_{l \in L_{ij}^{>}} \epsilon_j(\mathbf{x}^l) \;\leq\; e\ln(2\mu mn\rho) \max_{l \in L_{ij}^{>}} \mathrm{rate}_{ij}(\mathbf{x}^l)\,. \tag{29}$$

If $L_{ij}^{=} \neq \emptyset$, then during the execution of phases in $L_{ij}^{=}$, x_{ij} increases from an initial value of $z_i(j-1)$ to $z_i(j)$ after the completion of the phases in $L_{ij}^{=}$. If j is the first client in $C_i(Z)$, then its initial value is x_{ij}^0, hence $z_i(j-1) = x_{ij}^0$ as defined in (25). By a similar analysis as for $L_{ij}^{>}$,

$$\sum_{l \in L_{ij}^{=}} \epsilon_j^l \;\leq\; e\left(\ln\frac{z_i(j)}{z_i(j-1)}\right) \max_{l \in L_{ij}^{=}} \mathrm{rate}_{ij}(\mathbf{x}^l)\,. \tag{30}$$

The dual variable α_j gets incremented by $e\epsilon_j(\mathbf{x}^l)$ in each phase for client j. Hence, $\alpha_j = e\left(\sum_{l \in L_{ij}^{>}} \epsilon_j(\mathbf{x}^l) + \sum_{l \in L_{ij}^{=}} \epsilon_j(\mathbf{x}^l)\right)$. Thus, from (29) and (30),

$$\alpha_j \;\leq\; e^2\left(\ln(2\mu mn\rho) \max_{l \in L_{ij}^{>}} \mathrm{rate}_{ij}(\mathbf{x}^l) + \ln\frac{z_i(j)}{z_i(j-1)} \max_{l \in L_{ij}^{=}} \mathrm{rate}_{ij}(\mathbf{x}^l)\right)\,.$$

Replacing the values of $\mathrm{rate}_{ij}(\mathbf{x}^l)$ for $l \in L_{ij}^{>}$ and $l \in L_{ij}^{=}$ from (18),

$$\begin{aligned}\alpha_j \;\leq\; e^2 &\left(Z \max_{l \in L_{ij}^{>}} \frac{\partial \mathrm{est}(\mathbf{x}^l)}{\partial x_{ij}} + \frac{c_i p_{ij}}{Z\Gamma} + \frac{a_{ij}}{\Gamma}\right)\ln(2\mu mn\rho) \\ &+ \left(Z \max_{l \in L_{ij}^{=}} \frac{\partial \mathrm{est}(\mathbf{x}^l)}{\partial x_{ij}} + \frac{c_i p_{ij}}{Z\Gamma} + \frac{c_i}{\Gamma} + \frac{a_{ij}}{\Gamma}\right)\ln\frac{z_i(j)}{z_i(j-1)}\,.\end{aligned}$$

For any client j, $x_{ij} \le \mu$, and hence $z_i(j) \le \mu$. Since $x_{ij} \ge x^0_{ij}$, $z_i(j) \ge x^0_{ij}$. Thus, $z_i(j)/z_i(j-1) \le \mu/x^0_{ij} \le 2\mu mn\rho$. Thus

$$\alpha_j \;\le\; 2e^2 \left(Z \max_{l \in L_j} \frac{\partial \mathrm{est}(\mathbf{x}^l)}{\partial x_{ij}} + c_i \frac{p_{ij}}{Z\Gamma} + \frac{a_{ij}}{\Gamma} \right) \ln(2\mu mn\rho) + e^2 \frac{c_i}{\Gamma} \ln \frac{z_i(j)}{z_i(j-1)} \,,$$

and from the expression for $\partial \mathrm{est}(\mathbf{x}^l)/\partial x_{ij}$ in (24),

$$\begin{aligned} \alpha_j \;&\le\; 2e^2 \left(\frac{Z}{\Gamma} \left(\frac{p_{ij}}{Z} \eta_i + \chi_{ij} \right) + c_i \frac{p_{ij}}{Z\Gamma} + \frac{a_{ij}}{\Gamma} \right) \ln(2\mu mn\rho) + e^2 \frac{c_i}{\Gamma} \ln \frac{z_i(j)}{z_i(j-1)} \\ &=\; \frac{e^2}{\Gamma} \left[\left(p_{ij} \left(\eta_i + \frac{c_i}{Z} \right) + Z\chi_{ij} + a_{ij} \right) \frac{\sigma}{2e^2} + c_i \ln \frac{z_i(j)}{z_i(j-1)} \right] . \end{aligned} \tag{31}$$

By definition, $\beta_{ij} = Z\chi_{ij}\sigma/(2e^2) + c_i \ln \frac{z_i(j)}{z_i(j-1)}$, and $\gamma_i = \left(\eta_i + \frac{c_i}{Z}\right) \sigma/(2e^2)$. Replacing these expressions in (31) yields $\alpha_j \le \frac{e^2}{\Gamma} \left(\beta_{ij} + p_{ij}\gamma_i + a_{ij}\sigma/(2e^2) \right)$, proving the lemma. □

Lemma 23. *For all i,* $\displaystyle\sum_{j : i \in F_j(Z)} \beta_{ij} + Z\gamma_i - \delta_i \;\le\; c_i \frac{\sigma}{e^2}$.

Proof. By definition (28), for any i,

$$\sum_{j : i \in F_j(Z)} \beta_{ij} + Z\gamma_i - \delta_i \;=\; c_i \frac{\sigma}{2e^2} + c_i \sum_{j : i \in F_j(Z)} \ln \frac{z_i(j)}{z_i(j-1)} \;=\; c_i \frac{\sigma}{2e^2} + c_i \ln \frac{z_i(n)}{z_i(n_i - 1)}$$

where n is the total number of clients, and n_i is the index of the first client j such that $i \in F_j(Z)$. By definition (25), $z(n_i - 1) \ge 1/(2mn\rho)$, and $z_i(n) \le x_{ij} \le \mu$ by Fact 18. Hence

$$\sum_{j : i \in F_j(Z)} \beta_{ij} + Z\gamma_i - \delta_i \;\le\; 3c_i \ln(2\mu mn\rho) \;\le\; c_i \frac{\sigma}{e^2} \,.$$

□

Lemma 24. $\displaystyle\sum_i \delta_i \;\le\; Z(1 + \ln(mn) + \max_l \tilde{\lambda}(\mathbf{x}^l)) \frac{\sigma}{e^2}$.

Proof. We show that

$$\sum_{j, i \in F_j(Z)} \chi_{ij} \;\le\; 1 + \ln(mn) + \max_l \tilde{\lambda}(\mathbf{x}^l) \tag{32}$$

and

$$\sum_i \eta_i \;\le\; 1 + \ln(m) + \max_l \tilde{\lambda}(\mathbf{x}^l) \tag{33}$$

which suffices to prove the lemma. For (32), let $j \in [n]$ and $i \in F_j(Z)$, and define $\phi(i,j)$ as the phase that maximizes $\frac{e^{x^l_{ij}/\Gamma}}{\sum_{i',j'} e^{x^l_{i'j'}/\Gamma}}$. Define $b_{ij} := e^{x^{\phi(i,j)}_{ij}/\Gamma}$. Since variables are nondecreasing,

$$\sum_{j, i \in F_j(Z)} \max_{l \in L_j} \frac{e^{x^l_{ij}/\Gamma}}{\sum_{i',j'} e^{x^l_{i'j'}/\Gamma}} \;\le\; \sum_{j, i \in F_j(Z)} \frac{b_{ij}}{\sum_{(i',j') : \phi(i',j') \le \phi(i,j)} b_{i'j'}}$$

and by Lemma 44, this is at most $1 + \ln(\sum_{j,i\in F_j(Z)} b_{ij}) \le 1 + \ln(mn) + \max_l \tilde{\lambda}(\mathbf{x}^l)$.

Similarly, for (33), define $\phi(i)$ as the phase which maximizes $\frac{e^{\sum_j p_{ij}x^l_{ij}/(Z\Gamma)}}{\sum_{i'} e^{\sum_j p_{i'j}x_{i'j}/(Z\Gamma)}}$, and define $b_i := \exp\left(\sum_j p_{ij}x^{\phi(i)}_{ij}/(Z\Gamma)\right)$.

$$\sum_i \max_{l\in L} \frac{e^{\sum_j p_{ij}x^l_{ij}/(Z\Gamma)}}{\sum_{i'} e^{\sum_j p_{i'j}x^l_{i'j}/(Z\Gamma)}} \;\le\; \sum_i \frac{b_i}{\sum_{i':\phi(i')\le\phi(i)} b_{i'}}$$

and by Lemma 44, this is at most $1 + \ln m + \max_l \tilde{\lambda}(\mathbf{x}^l)$. □

Let $\nu := (1 + \ln(mn) + \max_l \tilde{\lambda}(\mathbf{x}^l))$. Then

Lemma 25. *For* $\mathbf{x}$ *obtained by* ASSIGN*(*Γ*)* , *the vectors* $\alpha' = \alpha/(\nu\sigma)$, $\beta' = e^2\beta/(\nu\sigma)$, $\gamma' = e^2\gamma/(\nu\sigma)$ *and* $\delta' = e^2\delta/(\nu\sigma)$ *are feasible for D2(*Z, Γ*).*

Proof. We show that the constraints in D2(Z, Γ) are satisfied by $(\alpha', \beta', \gamma', \delta')$. For the third constraint in D2(Z, Γ), from Lemma 24, $\sum_i \delta'_i = \sum_i \delta_i e^2/(\sigma\nu) \le Z$

For the second constraint,

$$\sum_j \beta'_{ij} + Z\gamma'_i - \delta'_i \;=\; \left(\sum_j \beta_{ij} + Z\gamma_i - \delta_i\right) e^2/(\nu\sigma) \;\le\; \left(\sum_j \beta_{ij} + Z\gamma_i - \delta_i\right) e^2/\sigma \;\le\; c_i$$

where the first inequality is because $\nu \ge 1$, and the last inequality is from Lemma 23.

For the first constraint,

$$\alpha'_j - \frac{1}{\Gamma}(\beta'_{ij} + p_{ij}\gamma'_i + a_{ij}) \;=\; \frac{\alpha_j}{\nu\sigma} - \frac{e^2}{\Gamma}\left(\frac{\beta_{ij}}{\nu\sigma} + p_{ij}\frac{\gamma_i}{\nu\sigma}\right) - \frac{a_{ij}}{\Gamma} \;\le\; 0\,,$$

where the last inequality follows from Lemma 22 and since $\nu \ge 1/2$. Hence, $(\alpha', \beta', \gamma', \delta')$ are feasible for D2(Z, Γ). □

We now use the primal-dual framework to show that given Z, Γ such that $4\sigma\frac{\mathcal{OPT}_1(Z)}{Z} \ge \Gamma \ge 2\sigma\frac{\mathcal{OPT}_1(Z)}{Z}$, our algorithm will succeed, and bound the competitive ratio obtained for LP1(Z).

Lemma 26. *If CCFL-LP1(Z) is feasible and* $\frac{\Gamma}{2\sigma} \ge \frac{\mathcal{OPT}_1(Z)}{Z}$*, then* ASSIGN*(*$\Gamma$*) does not fail.*

Proof. Let $(\tilde{\mathbf{x}}^f, \tilde{\mathbf{y}}^f, \tilde{\lambda}^f)$ be the current values for LP2(Z, Γ) and $(\alpha^f, \beta^f, \gamma^f, \delta^f)$ be the current values for D2(Z, Γ) when the condition in line 10 is being checked. Let $(\tilde{\mathbf{x}}^*, \tilde{\mathbf{y}}^*, \tilde{\lambda}^*)$ and $(\alpha^*, \beta^*, \gamma^*, \delta^*)$ be the optimal primal and dual solutions for LP2(Z, Γ) and D2(Z, Γ). Then

$$\mathcal{OPT}_2(Z, \Gamma) \;=\; Z\tilde{\lambda}^* + \sum_i c_i\tilde{y}^*_i + \sum_{i,j} a_{ij}x^*_{ij}/\Gamma \;=\; \sum_j \alpha^*_j \tag{34}$$

where the second equality is because of LP strong duality. From Lemma 25, $\alpha^f/(\nu\sigma)$ is feasible for the dual. Hence $\sum_j \alpha^*_j \ge \sum_j \alpha^f_j/(\nu\sigma)$. Then from (34), and from Fact 17,

$$\mathcal{OPT}_1(Z) \;=\; \Gamma\mathcal{OPT}_2(Z, \Gamma) \;\ge\; \Gamma\sum_j \alpha^f_j/(\nu\sigma)\,.$$

Since $\sum_j \alpha_j^f$ is an upper bound on the change in $\mathrm{cost}(\mathbf{x})$ in each phase, $\sum_j \alpha_j^f \geq \mathrm{cost}(\mathbf{x}^f) - \mathrm{cost}(\mathbf{0}) - \sum_j \mathrm{init}_j$. Thus

$$\nu\sigma\mathcal{OPT}_1(Z) \;\geq\; \Gamma \left(\mathrm{cost}(\mathbf{x}^f) - \mathrm{cost}(\mathbf{0}) - \sum_j \mathrm{init}_j \right) .$$

$\mathcal{OPT}_1(Z)/Z \geq 1$ by Fact 16. Hence $\Gamma \geq 1$ by the condition in the lemma statement, and thus from Lemma 19, and since $\mathrm{cost}(\mathbf{0}) \leq 2Z\ln(mn)$,

$$\nu\sigma\mathcal{OPT}_1(Z) \;\geq\; \Gamma \left(\mathrm{cost}(\mathbf{x}^f) - 2Z - 2Z\ln(mn) \right) .$$

By definition, $\nu = 1 + \ln(mn) + \tilde{\lambda}(\mathbf{x}^f)$, and $\tilde{\lambda}(\mathbf{x}^f) \leq \mathrm{cost}(\mathbf{x}^f)/Z$. With these substitutions,

$$\sigma\mathcal{OPT}_1(Z) \left(\ln(emn) + \frac{\mathrm{cost}(\mathbf{x}^f)}{Z} \right) \;\geq\; \Gamma \left(\mathrm{cost}(\mathbf{x}^f) - 2Z - 2Z\ln(mn) \right) , \tag{35}$$

and from the bound on Γ in the lemma statement,

$$\frac{1}{2} Z \left(\ln(emn) + \frac{\mathrm{cost}(\mathbf{x}^f)}{Z} \right) + 2Z\ln(mn) + 2Z \;\geq\; \mathrm{cost}(\mathbf{x}^f) .$$

Simplifying yields $\frac{5}{2} Z\ln(emn) \geq \frac{1}{2}\mathrm{cost}(\mathbf{x}^f)$. Hence, $\mathrm{cost}(\mathbf{x}^f) \leq 5Z\ln(emn)$ and the algorithm will not fail. □

Lemma 27. *If CCFL-LP1(Z) is feasible and* $\frac{\mathcal{OPT}_1(Z)}{Z} \leq \frac{\Gamma}{2\sigma} \leq 2\frac{\mathcal{OPT}_1(Z)}{Z}$, ASSIGN*($\Gamma$) returns a solution to CCFL-LP1(Z) of cost* $O(\ln(mn)\ln(mn\rho))\mathcal{OPT}_1(Z)$.

Proof. Let $(\mathbf{x}^f, \tilde{\mathbf{y}}^f, \tilde{\lambda}^f)$ be the solution for LP2(Z, Γ) our algorithm returns. By Lemma 26, ASSIGN(Γ) does not fail, and hence $\mathrm{cost}(\mathbf{x}^f) \leq 5Z\ln(emn)$. Substituting this bound on $\mathrm{cost}(\mathbf{x}^f)$ in the expression on the left in (35) yields

$$\sigma\mathcal{OPT}_1(Z)\, 6\ln(emn) \;\geq\; \Gamma \left(\mathrm{cost}(\mathbf{x}^f) - 2Z - 2Z\ln(mn) \right) .$$

or $6\ln(emn)\sigma\mathcal{OPT}_1(Z) + 2Z\Gamma\ln(emn) \geq \Gamma\mathrm{cost}(\mathbf{x}^f)$. Substituting the upper bound on Γ,

$$6\sigma\ln(emn)\mathcal{OPT}_1(Z) + 8\sigma\ln(emn)\mathcal{OPT}_1(Z) \;\geq\; \Gamma\mathrm{cost}(\mathbf{x}^f) .$$

Since $\Gamma\mathrm{cost}(\mathbf{x}^f)$ is an upper bound on the cost of the solution obtained for CCFL-LP1(Z), the proof follows. □

The following theorem now follows immediately from Lemmas 26 and 27.

Theorem 28. *If CCFL-LP1(Z) is feasible and* Z, Γ *satisfy* $2\frac{\mathcal{OPT}_1(Z)}{Z} \geq \frac{\Gamma}{2\sigma} \geq \frac{\mathcal{OPT}_1(Z)}{Z}$, *then* ASSIGN*($\Gamma$) does not fail and returns a solution to CCFL-LP1(Z) of cost* $O(\ln(mn)\ln(mn\rho))\mathcal{OPT}_1(Z)$.

3.2 A Doubling Procedure for Γ.

If we are not given Γ that satisfies the conditions of Theorem 28, we use a doubling procedure similar to that described in Section 2.2. Initially set $\Gamma = 1$ and run ASSIGN(Γ) . Each execution of ASSIGN(Γ) is called a *trial*, and each trial τ has a distinct set of primal and dual variables ($\tilde{\mathbf{x}}(\tau)$, $\tilde{\mathbf{v}}(\tau)$, $\tilde{\mathbf{w}}(\tau)$, $\tilde{\lambda}(\tau)$). In each trial, $x_{ij}(\tau)$ is initialized to x^0_{ij} for each client that arrives during that trial, and the other variables are updated accordingly. If a trial fails, we double Γ and proceed with a new trial with a new set of primal and dual variables. We continue in this manner, doubling the value of Γ after each failure, until all clients are assigned. The cost of the solution we obtain is then at most the sum of the costs obtained in each trial.

Let $\tilde{\mathcal{A}}(Z, \Gamma)$ be the cost of the solution to LP2(Z, Γ) obtained in each trial. Hence, by Fact 17, $\Gamma\tilde{\mathcal{A}}(Z, \Gamma)$ is the cost of the solution to CCFL-LP1(Z) in each trial. Define $\mathcal{A}(Z)$ as the sum of the (partial) solutions to CCFL-LP1(Z) in each trial. Thus, $\mathcal{A}(Z) := \sum_\Gamma \Gamma\tilde{\mathcal{A}}(Z, \Gamma)$.

Theorem 29. *If CCFL-LP1(Z) is feasible,* ASSIGN*(Γ) with the doubling procedure for Γ obtains a fractional solution to CCFL-LP1(Z) of cumulative cost $O(\ln(mn)\ln(mn\rho))\mathcal{OPT}_1(Z)$ over all trials.*

Proof. By assumption, CCFL-LP1(Z) is feasible. Initially, $\Gamma = 1 \leq \mathcal{OPT}_1(Z)/Z$ since $\lambda \geq 1$. Since we double Γ each time ASSIGN(Γ) fails, and by Lemma 26 ASSIGN(Γ) will not fail for $\Gamma \geq 2\sigma\mathcal{OPT}_1(Z)/Z$, $\Gamma \leq 4\sigma\mathcal{OPT}_1(Z)/Z$ in any trial. Further, in any successful trial, $\tilde{\mathcal{A}}(Z, \Gamma) \leq 8Z\ln(emn)$. For any failed trial τ, at the beginning of the phase when the trial failed, $\tilde{\mathcal{A}}(Z, \Gamma) \leq \text{cost}(\mathbf{x}) \leq 6Z\ln(3mn)$ by Corollary 20. In one phase, each variables x_{ij} is incremented by at most a factor of $\mu = 1+1/(6\ln(emn))$. Thus in any failed trial, $\tilde{\mathcal{A}}(Z, \Gamma) \leq \mu 6Z\ln(emn) \leq Z(1 + 6\ln(emn))$.

Thus over all trials, the cumulative cost of the solution to CCFL-LP1(Z) $\mathcal{A}(Z)$ is at most $Z(1 + 6\ln(emn))\sum_\Gamma \Gamma$. Let Γ^f be the value of Γ in the final trial. Since Γ is doubled after each trial, $\mathcal{A}(Z) \leq Z(1 + 6\ln(emn))2\Gamma^f$. Since $\Gamma^f \leq 4\sigma\mathcal{OPT}_1(Z)/Z$, and $\sigma = 4e^2\ln(2\mu mn\rho)$, the theorem follows. □

3.3 Obtaining an Integral Solution

We will now build upon the fractional algorithm for CCFL and give a randomized rounding procedure that obtains an integral assignment of clients to facilities. As before, we will assume that the fractional assignment x_{ij} of client j to any facility i not in $F_j(Z)$ is always zero. We first give a rounding procedure that uses Theorem 29 and obtains an integral assigment of clients to facilities. We will then show how to use this integral assignment procedure for a fixed parameter Z to obtain an $O(\ln^2(mn)\ln(mn\rho))$-competitive solution to Z^*.

We run our rounding procedure whenever a new client j arrives, and given a fractional solution $(\mathbf{x}, \mathbf{y}, \lambda)$ for CCFL-LP1(Z) that satisfies $\sum_i x_{ij} \geq 1$. The procedure returns a set of open facilities and an integral assignment of j to an open facility.

We assume $x_{ij} \leq 1$ without loss of generality. For each client j, let $S(j) := \{i : x_{ij} \geq 1/(2m)\}$. C_j is a set of *candidate* facilities for the integral assignment for client j. Initially, $C_j = \emptyset$. $\mathcal{O}$ is the set of facilities opened so far, and $\mathcal{O} = \emptyset$ initially.

Our randomized rounding procedure is as follows. For each facility i, select $r = \lceil 4e\ln n\rceil$ random variables uniformly at random between 0 and 1; let t_{i1}, t_{i2}, …, $t_{i,r}$ be these random variables for facility i, and $\bar{t}_i := \min_k t_{ik}$. When client j arrives,

Step 1: For each facility $i \notin \mathcal{O}$, add i to $\mathcal{O}$ if $y_i \geq \bar{t}_i$.

Step 2: For each facility $i \in S_j$, add i to C_j independently with probability x_{ij}/y_i. If $i \in C_j$, then i is a *candidate* for j.

Step 3: For each facility $i \in \mathcal{O}$, assign client j to i if i is a candidate for j. Denote an assignment of client j to facility i by $j \to i$. If j is assigned to multiple facilities, pick one arbitrarily.

Step 4: If j is not yet assigned to any facility, assign it to facility $i \in S_j$ that minimizes $c_i + a_{ij} + p_{ij}$.

Steps 3 and 4 thus give an integral assignment of clients to facilities. We show:

Theorem 30. *The integral assignment obtained has total cost* $O(\ln(mn)\ln(mn\lambda)\ln(mn\rho))\mathcal{OPT}(Z)$.

We first show that with high probability, no client is assigned in Step 4:

Lemma 31. *For any client, the probability that it is assigned in Step 3 is at least* $1 - 1/n^2$.

Proof. For a client j that has just arrived, consider a facility $i \in S_j$. In Step 3,

$$\begin{aligned}
\Pr[j \to i] &= \Pr[i \text{ is open}] \cdot \Pr[i \in C_j] \\
&= (1 - \Pr[y_i < \bar{t}_i]) \, \frac{x_{ij}}{y_i} \\
&= (1 - (1 - y_i)^r) \, \frac{x_{ij}}{y_i} \\
&\geq (1 - e^{-y_i r}) \, \frac{x_{ij}}{y_i} \\
&\geq r x_{ij}/e
\end{aligned}$$

where the first inequality is because $(1 + x) \leq e^x$ for all $x \in \mathbb{R}$, and the second inequality is because $e^{-x} \leq 1 - (x/e)$ for $0 \leq x \leq 1$. Thus, the probability that j is not assigned to a fixed $i \in S_j$ is at most $1 - (rx_{ij}/e) \leq e^{-rx_{ij}/e}$. Since these probabilities are independent,

$$\begin{aligned}
\Pr[j \text{ is not assigned}] &= \prod_{i \in S_j} \Pr[j \text{ is not assigned to } i] \\
&\leq \prod_{i \in S_j} e^{-rx_{ij}/e} = e^{-r\sum_{i \in S_j} x_{ij}/e}.
\end{aligned}$$

For any $i \notin S_j$, $x_{ij} \leq 1/(2m)$, hence $\sum_{i \notin S_j} x_{ij} \leq 1/2$. Thus $\sum_{i \in S_j} x_{ij} \geq 1/2$, and since $r \geq 4e \ln n$, the probability that client j is not assigned to any facility in Step 3 is at most $1/(n^2)$. □

Since each client is assigned in Step 3 with high probability, the effect of Step 4 on the total cost of the integral assignment is negligible. The following lemma follows immediately from Lemma 31 and since $S_j \subseteq F_j(Z)$:

Lemma 32. *The assignments in Step 4 increase the total cost of the integral assignment by at most* Z/n.

We now show bounds on the total cost for assignments in the remaining steps. We first bound the expected fixed-charges and assignment costs.

Lemma 33. *The expected sum of fixed-charges* $\sum_{i \in \mathcal{O}} c_i$ *is at most* $r \sum_i c_i y_i$.

Proof. The probability that facility i is in $\mathcal{O}$ is $\Pr[y_i \geq \bar{t}_i] \leq \sum_{k=1}^{r} \Pr[y_i \geq t_{ik}] = ry_i$ by the union bound. The lemma follows. □

Lemma 34. *The expected assignment costs for clients assigned in Step 3 is at most* $r \sum_{i,j} a_{ij} x_{ij}$.

Proof. For facility i, the probability that i is in $\mathcal{O}$ is at most ry_i by the proof of Lemma 33. For any client j assigned in Step 3, $\Pr[j \to i] = \Pr[i \text{ is open}] \cdot \Pr[i \in C_j] \leq rx_{ij}$. The bound on the expected assignment cost follows. □

We now prove the bound on the expected maximum congestion for the integral assignment. Define the *candidate congestion* for a facility i as $L_i^{(c)} := \sum_{j:i\in C_j} p_{ij}$. For any realization of the random bits, the candidate congestion of any facility is an upper bound on the actual congestion for clients assigned to the facility in Step 3. We will prove an upper bound on the expected value of the maximum candidate congestion over all facilities, which will give us a bound on the expected value of the maximum congestion.

We consider the maximum candidate congestion instead of the actual maximum congestion because for a fixed facility i, the actual assignments of the clients are not independent of each other. If client $j-1$ is assigned to facility i, then the facility must be open, and hence the probability that client j is assigned to i increases. However, for any facility i and clients $j \neq j'$, Pr[$i \in C_j$] and Pr[$i \in C_{j'}$] are independent.

For the next lemma, for any client j that arrived in the current trial, define $y_i(j)$ as the value of y_i when the randomized rounding procedure was executed for client j.

Lemma 35. *The candidate congestion $L_i^{(c)}$ on any facility i at most $Z\ln(2em\lambda)$ in expectation.*

Proof. Consider a fixed open facility i.

$$E[L_i^{(c)}] \;=\; \sum_{j:i\in S_j} \frac{p_{ij}x_{ij}}{y_i(j)}\,.$$

For any client j,

$$Z \;\geq\; \frac{1}{y_i(j)} \sum_{j'\leq j} p_{ij'}x_{ij'}\,. \tag{36}$$

By Lemma 47 with $P = \sum_{j:i\in S_j} \frac{p_{ij}x_{ij}}{y_i(j)}$ and Z as defined here,

$$E[L_i^{(c)}] \;=\; P \;\leq\; Z\left(1+\ln\frac{y_i(n)}{y_i(k)}\right)\,.$$

where k is the first client j such that $i \in S_j$. Then $y_i(k) \geq x_{ik} \geq 1/(2m)$, and $y_i(n) \leq \lambda$. The lemma follows. □

To bound the maximum candidate congestion, we use the following inequality:

Lemma 36 ([23])**.** *Let $X_1,\ldots,X_n$ be independent random variables with Pr($X_j = 1$) $= q_j$, Pr($X_j = 0$) $= 1-q_j$. For $X = \sum_{j=1}^n a_jX_j$, define $\nu = \sum_{j=1}^n a_j^2q_j$ and $a = \max_j a_j$. Then*

$$\textit{Pr}(X > E(X)+\mu) \leq e^{-\frac{\mu^2}{2\nu+\frac{2a\mu}{3}}}$$

Lemma 37. *The maximum candidate congestion is at most $4Z\ln(2em\lambda)$ in expectation.*

Proof. Fix facility i. For each client j, let $a_j = p_{ij}/Z$ if $i \in S_j$, and $a_j = 0$ otherwise. Hence $a_j \leq 1$. Define random variable $X_j = 1$ if $i \in C_j$, and $X := \sum_j a_jX_j = \frac{L_i^{(c)}}{Z}$, since $C_j \subseteq S_j$. Let $q_j :=$ Pr[$i \in C_j$]. From Lemma 35, $E(X) \leq \ln(2em\lambda)$.

We will use Lemma 36 to show that with high probability, the candidate congestion $L_i^{(c)}$ on facility i does not exceed thrice the expected value. Let $\mu := 3\ln(2em\lambda)$. Then from Lemma 36,

$$\Pr\left(X > 3\ln(2em\lambda)\right) \;\leq\; e^{-\frac{\mu^2}{2\nu+\frac{2a\mu}{3}}}$$

Since $a_j \leq 1$ for all j, $a \leq 1$. Also, $\nu = \sum_j a_j^2 q_j \leq \sum_j a_j q_j = E(X) = \mu/3$. Then

$$\begin{aligned} \Pr\left(X > 3\ln(2em\lambda)\right) &\leq e^{-\frac{\mu^2}{\frac{2\mu}{3}+\frac{2\mu}{3}}} \\ &\leq e^{-\frac{3\mu}{4}} \\ &\leq (2em\lambda)^{-2} \leq \frac{1}{4m^2} \end{aligned} \tag{37}$$

where the last inequality is because $\lambda \geq 1$ by CCFL-LP1(Z). Thus, the probability that the candidate congestion of a fixed facility exceeds $3Z\ln(2em\lambda)$ is at most $1/(4m^2)$, and by the union bound, the probability that the candidate congestion of any facility exceeds $3Z\ln(2em\lambda)$ is at most $1/(4m)$. To get the bound on the expectation of the maximum candidate congestion, we observe that the candidate congestion of any facility can at most be $\sum_{j:i\in S_j} p_{ij} \leq 2m\lambda \sum_j p_{ij}x_{ij}/y_i(j)$, since $x_{ij} \geq 1/(2m)$ for any client j with $i \in S_j$, and $y_i(j) \leq \lambda$. From the constraints in CCFL-LP1(Z), $\sum_j p_{ij}x_{ij}/y_i(j) \leq Z$, and hence the candidate congestion is bounded by $2m\lambda Z$. Thus the expected value of the maximum candidate congestion is at most $3Z\ln(2em\lambda)(1 - 1/(4e^2m^2\lambda^2)) + 2m\lambda Z/(4e^2m^2\lambda^2) \leq 4Z\ln(2em\lambda)$ using $\lambda \geq 1$. □

We now use the bounds on the congestion, fixed-charges and assignment costs to prove the bound on the expected total cost from Theorem 30.

Proof of Theorem 30. By Theorem 28 and Lemmas 33 and 34, the sum of the fixed-charges and assignment costs for the integral assignments is $O(\ln n \ln(mn)\ln(mn\rho))\ \mathcal{OPT}(Z)$ in expectation. By Lemma 37, the maximum congestion is $O(\ln(m\lambda))Z \leq O(\ln m)\mathcal{OPT}_1(Z)$ in expectation, since $\mathcal{OPT}_1(Z) \geq Z\lambda$. Assignments in Step 4 add at most Z/n to the total cost by Lemma 32. Summing up, the total cost of the integral assignment is $O(\ln(mn\lambda)\ln(mn)\ln(mn\rho))\ \mathcal{OPT}_1(Z)$ in expectation. □

3.4 A Doubling Procedure for Z

We now use the rounding procedure with a doubling argument and describe an algorithm for the CCFL problem. We assume we are given a procedure that, given Z, maintains an integral assigment of clients to facilities of total cost $O(\ln^2(mn)\ln(mn\rho))\ \mathcal{OPT}_1(Z)$. Let $\mathbb{Q}(Z)$ denote this procedure, and let $c_{\mathbb{Q}}(Z)$ be the expected total cost obtained by this procedure. We start with $Z = \min_i\{c_i + p_{ij} + a_{ij}\}$, and run $\mathbb{Q}(Z)$. Call each execution of $\mathbb{Q}(Z)$ an *epoch*. In each epoch, $\mathbb{Q}(Z)$ uses a distinct set of variables. If $c_{\mathbb{Q}}(Z)$ $\geq O(\ln^2(mn)\ln(mn\rho))Z$ in an epoch, or if for some client j $F_j(Z) = \emptyset$, we fail $\mathbb{Q}(Z)$, double the value of Z, and run $\mathbb{Q}(Z)$ for the clients not yet assigned with the new value of Z and a new set of variables.

We start by showing that if Z is at least Z^*, then Z and $\mathcal{OPT}(Z)$ are close, and bounding the total cost returned by the procedure $\mathbb{Q}(Z)$.

Lemma 38. *If $Z \geq Z^*$, then $\mathcal{OPT}(Z) \leq 2Z$ and $c_{\mathbb{Q}}(Z) \leq O(\ln^2(mn)\ln(mn\rho))Z$.*

Proof. If $Z \geq Z^*$, then CCFL-LP1(Z) is feasible, since for every client j, $F_j(Z) \neq \emptyset$. Consider the solution to CCFL-LP1(Z) that sets $y_i = 1$ for every facility that is open in the optimal solution, and $x_{ij} = 1$ if client j is assigned to facility i in the optimal solution. Set $\lambda = 1$. Since Z^* is an upper bound on the congestion of any facility in the optimal solution and $Z \geq Z^*$, this gives a feasible solution to CCFL-LP1(Z). Then the sum of the assignment costs and fixed charges for this solution are at most Z^*. Also, $Z\lambda = Z$. Hence, the total cost of this solution is at most $Z + Z^* \leq 2Z$, and hence $\mathcal{OPT}_1(Z) \leq 2Z$. The bound on $c_{\mathbb{Q}}(Z)$ follows from Theorem 30. □

Finally, using $\mathbb{Q}(Z)$ with the doubling argument described,

Theorem 39. *The sum over all epochs of the expected total cost in each epoch is* $O(\ln^2(mn)\ln(mn\rho))Z^*$.

Proof. Let Z^f be the value of Z in the final epoch. By Lemma 38 and our failure condition for an epoch, if $Z^f \geq Z^*$ then the epoch must succeed. Since Z is doubled after every failed epoch, $Z^f \leq 2Z^*$. For every epoch, $c_{\mathbb{Q}}(Z) \leq O(\ln^2(mn)\ln(mn\rho))Z$, since otherwise we would have failed the current epoch. Since we double Z on every failed epoch, the sum over all epochs of the total cost in each epoch is $\sum_Z c_{\mathbb{Q}}(Z)$ $\leq O(\ln^2(mn)\ln(mn\rho))\sum_Z Z$, where the sum is over the values of Z in each epoch. Since Z is doubled after each epoch, $\sum_Z Z \leq 2Z^f \leq 4Z^*$. Hence, the total cost over all epochs of the integral assignment obtained by our algorithm is $O(\ln^2(mn)\ln(mn\rho))Z^*$. □

3.5 A Lower Bound for UMSC and CCFL

In this section we give a lower bound on the competitive ratio for bicriteria results for UMSC. These lower bounds on bicriteria results motivate the problem of minimizing the sum of makespan and startup costs for machine scheduling that we study in Section 3.

CCFL generalizes UMSC, and our lower bound extends to bicriteria results for CCFL as well. Let T^* be the makespan of an assignment of jobs to machines, m and n be the number of machines and jobs respectively, and ρ as defined in Section 3. Let C^* be the optimal startup cost of an assignment with makespan T^*. We show

Theorem 40. *No deterministic online algorithm can obtain a solution with makespan* $o(m)T^*$ *and startup cost within a factor polylogarithmic in* m, n, *and* ρ *of* C^*, *even if* T^* *is available offline.*

Our lower bound is in fact for any fractional solution that allows jobs to be assigned fractionally to machines, and machines to be fractionally opened. For each job, the sum of the fractional assigments to machines must be at least 1, and the fraction by which any machine is opened must be at least the fractional assignment of any job to the machine.

Our example has m machines, and $2(m-1)$ jobs. We choose $T^* = m$. The sequence of job arrivals is fixed, and known to the online algorithm; the only freedom we allow is that we could stop sending jobs from the sequence at any time, and send trivial jobs with $p_{ij} = 0$ for all i instead. We will show that no deterministic online algorithm can obtain a startup cost that exceeds the optimal startup cost by factor at most polynomial in m, and a makespan at most $T^*m/2$, even in a fractional solution.

The cost of machine i in our example is $e^{m(i-1)}$. Thus machine 1 has cost 1, and machine m has cost e^{m^2-m}. The index of each job corresponds to its arrival in the sequence. Thus job j, if it arrives, is the jth job to arrive. The jobs are of two types, *even* jobs and *odd* jobs, corresponding to their index. An odd job j can be assigned to either of two machines: machine 1 with processing time T^*, or machine $(j+3)/2$ with processing time ϵ. An even job j can only be assigned to machine $(j+2)/2$.

We start with an observation about the optimal assignment.

Lemma 41. *For jobs* $1, \ldots, k$ *with* k *even, there is an assignment of these jobs on machines* $2, \ldots, (k+2)/2$ *with makespan* T^*.

Proof. Assign each odd job j to machine $(j+3)/2$, and each even job j to machine $(j+2)/2$. Then no job is assigned to machine 1, and no job gets assigned to machine i for $i > (k+2)/2$. Also, each machine $2 \leq i \leq (k+2)/2$ gets assigned at most one odd job with processing time ϵ, and one even job with processing time $T^* - \epsilon$. The lemma follows. □

Suppose now that an odd job j arrives. Then

Machines	1	2	3	4	5
Job 1	T^*	ϵ			
Job 2		$T^* - \epsilon$			
Job 3	T^*		ϵ		
Job 4			$T^* - \epsilon$		
Job 5	T^*			ϵ	
Job 6				$T^* - \epsilon$	
Job 7	T^*				ϵ
Job 8					$T^* - \epsilon$

Table 1: The example depicted for 5 machines. All missing entries are ∞.

Lemma 42. *Any fractional algorithm that obtains startup cost $o(e^m/m)$ of the optimal startup cost must assign job j to machine 1 with fractional value $\geq 1/2$.*

Proof. Let $k := (j+3)/2$. By Lemma 41, jobs $1, \ldots, j-1$ can be assigned to machines $2, \ldots, k-1$ with makespan T^*. Job j can be assigned to either machine 1 with processing time T^*, or machine k with processing time ϵ. Thus in an optimal assignment of jobs $1, \ldots, j$, these jobs can be assigned to machines $1, \ldots, k-1$ with makespan T^*. The online algorithm cannot assign j to machine k with fractional value $\geq 1/2$, since the startup cost would exceed $e^{km}/2$, while the startup cost for the optimal assignment is at most $me^{(k-1)m}$. Thus, the algorithm must assign j to machine 1 with fractional value $\geq 1/2$. □

From Lemma 42, every odd job must be assigned to machine 1 with fractional value $\geq 1/2$, and hence the makespan obtained must be at least $m/2$, since in our example we send $2(m-1)$ jobs. Thus, any deterministic algorithm must have makespan at least $m/2$, or startup cost $\Omega(e^m/m)$.

Acknowledgements. We thank Zhenghui Wang for helpful discussions regarding various objectives for the machine scheduling problem, and for the example which appears in Section 3.5.

References

[1] Noga Alon, Baruch Awerbuch, Yossi Azar, Niv Buchbinder, and Joseph Naor. A general approach to online network optimization problems. *ACM Transactions on Algorithms*, 2(4):640–660, 2006.

[2] Noga Alon, Baruch Awerbuch, Yossi Azar, Niv Buchbinder, and Joseph Naor. The online set cover problem. *SIAM J. Comput.*, 39(2):361–370, 2009.

[3] Noga Alon, Yossi Azar, and Shai Gutner. Admission control to minimize rejections and online set cover with repetitions. *ACM Transactions on Algorithms*, 6(1), 2009.

[4] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta algorithm and applications. Technical report, 2005.

[5] Nikhil Bansal, Niv Buchbinder, and Joseph Naor. A primal-dual randomized algorithm for weighted paging. In *FOCS*, pages 507–517, 2007.

[6] Nikhil Bansal, Niv Buchbinder, and Joseph Naor. Randomized competitive algorithms for generalized caching. In *STOC*, pages 235–244, 2008.

[7] Daniel Bienstock and Garud Iyengar. Approximating fractional packings and coverings in o(1/epsilon) iterations. *SIAM J. Comput.*, 35(4):825–854, 2006.

[8] Ken Birman, Gregory Chockler, and Robbert van Renesse. Toward a cloud computing research agenda. *SIGACT News*, 40(2):68–80, 2009.

[9] Niv Buchbinder, Kamal Jain, and Joseph Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. In *ESA*, pages 253–264, 2007.

[10] Niv Buchbinder and Joseph Naor. Online primal-dual algorithms for covering and packing problems. In *ESA*, pages 689–701, 2005.

[11] Niv Buchbinder and Joseph Naor. The design of competitive online algorithms via a primal-dual approach. *Foundations and Trends in Theoretical Computer Science*, 3(2-3):93–263, 2009.

[12] Niv Buchbinder and Joseph Naor. Online primal-dual algorithms for covering and packing. *Math. Oper. Res.*, 34(2):270–286, 2009.

[13] Nikhil R. Devanur. Online algorithms with stochastic input. *SIGecom Exchanges*, 10(2):40–49, 2011.

[14] György Dósa and Zhiyi Tan. New upper and lower bounds for online scheduling with machine cost. *Discrete Optimization*, 7(3):125–135, 2010.

[15] Jon Feldman, Monika Henzinger, Nitish Korula, Vahab S. Mirrokni, and Clifford Stein. Online stochastic packing applied to display ad allocation. In *ESA (1)*, pages 182–194, 2010.

[16] Lisa Fleischer. Approximating fractional multicommodity flow independent of the number of commodities. *SIAM J. Discrete Math.*, 13(4):505–520, 2000.

[17] Lisa Fleischer. Data center scheduling, generalized flows, and submodularity. In *ANALCO*, pages 56–65, 2010.

[18] Dimitris Fotakis. On the competitive ratio for online facility location. *Algorithmica*, 50(1):1–57, 2008.

[19] Naveen Garg and Jochen Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. *SIAM J. Comput.*, 37(2):630–652, 2007.

[20] Michael D. Grigoriadis and Leonid G. Khachiyan. Fast approximation schemes for convex programs with many blocks and coupling constraints. *SIAM Journal on Optimization*, 4(1):86–107, 1994.

[21] Michael D. Grigoriadis and Leonid G. Khachiyan. An exponential-function reduction method for block-angular convex programs. *Networks*, 26(2):59–68, 1995.

[22] Michael D. Grigoriadis and Leonid G. Khachiyan. Approximate minimum-cost multicommodity flows in $\tilde{o}(\epsilon^{-2}knm)$ time. *Math. Program.*, 75:477–482, 1996.

[23] Michel Habib, Colin McDiarmid, Jorge Ramirez-Alfonsin, and Bruce. Reed, editors. *Probabilistic Methods for Algorithmic Discrete Mathematics*, volume 16 of *Algorithms and Combinatorics*. Springer-Verlag, Berlin, 1998.

[24] Csanád Imreh. Online scheduling with general machine cost functions. *Discrete Applied Mathematics*, 157(9):2070–2077, 2009.

[25] Csanád Imreh and John Noga. Scheduling with machine cost. In *RANDOM-APPROX*, pages 168–176, 1999.

[26] Samir Khuller, Jian Li, and Barna Saha. Energy efficient scheduling via partial shutdown. In *SODA*, pages 1360–1372, 2010.

[27] Philip N. Klein, Serge A. Plotkin, Clifford Stein, and Éva Tardos. Faster approximation algorithms for the unit capacity concurrent flow problem with applications to routing and finding sparse cuts. *SIAM J. Comput.*, 23(3):466–487, 1994.

[28] Christos Koufogiannakis and Neal E. Young. Beating simplex for fractional packing and covering linear programs. In *FOCS*, pages 494–504, 2007.

[29] Frank Thomson Leighton, Fillia Makedon, Serge A. Plotkin, Clifford Stein, Éva Stein, and Spyros Tragoudas. Fast approximation algorithms for multicommodity flow problems. *J. Comput. Syst. Sci.*, 50(2):228–243, 1995.

[30] Mohammad Mahdian, Yinyu Ye, and Jiawei Zhang. Approximation algorithms for metric facility location problems. *SIAM J. Comput.*, 36(2):411–432, 2006.

[31] Serge A. Plotkin, David B. Shmoys, and Éva Tardos. Fast approximation algorithms for fractional packing and covering problems. In *FOCS*, pages 495–504, 1991.

[32] Asfandyar Qureshi, Rick Weber, Hari Balakrishnan, John Guttag, and Bruce Maggs. Cutting the electric bill for internet-scale systems. In *ACM SIGCOMM*, Barcelona, Spain, August 2009.

[33] Farhad Shahrokhi and David W. Matula. The maximum concurrent flow problem. *J. ACM*, 37(2):318–334, 1990.

[34] Neal E. Young. Sequential and parallel algorithms for mixed packing and covering. In *FOCS*, pages 538–546, 2001.

[35] Jiawei Zhang, Bo Chen, and Yinyu Ye. A multiexchange local search algorithm for the capacitated facility location problem. *Math. Oper. Res.*, 30(2):389–403, 2005.

A Appendix

We present technical lemmas which are used in various proofs.

Lemma 43. *Let $y := \sum_{i=1}^{n} r_i$, where $0 < r_i \leq 1$ for $i \in [n]$, and $\prod_{i=1}^{n} r_i = P$. Then y is minimized when $r_i = P^{1/n}\ \forall i$, and the minimum value is $nP^{1/n}$.*

Proof. The proof is by induction on n. For $n = 1$, the lemma is obviously true. Let $\gamma_k(P)$ be the minimum value of the sum of k variables, when the product of the variables is P. Then $\gamma_n(P) = \min_{0<r_1\leq 1}\{r_1 + \gamma_{n-1}(P/r_1)\}$. By the inductive hypothesis, $\gamma_{n-1}(P/r_1) = (n-1)(P/r_1)^{1/(n-1)}$. Hence

$$\gamma_n(P) \;=\; \min_{0<r_1\leq 1}\left\{r_1 + (n-1)\left(\frac{P}{r_1}\right)^{1/(n-1)}\right\}.$$

We will show that the expression on the right is minimized when $r_1 = P^{1/n}$. Then by the inductive hypothesis, each of the other variables is $P^{1/n}$ as well, completing the proof.

Let $z = r_1 + (n-1)\left(\frac{P}{r_1}\right)^{1/(n-1)}$. Then

$$\frac{dz}{dr_1} \;=\; 1 - \frac{1}{n-1} r_1^{-n/(n-1)} (n-1) P^{1/(n-1)}\,,$$

and setting $dz/dr_1 = 0$, we obtain $r_1 = P^{1/n}$. Further, $d^2z/dr_1^2 \geq 0\ \forall r_1 \geq 0$. Hence, the point $r_1 = P^{1/n}$ is a minimum. This completes the proof. □

Lemma 44. *For any $n \in \mathbb{Z}_+$ and $a_1, a_2, \ldots, a_n \in \mathbb{R}_{\geq 0}$ with $a_1 > 0$,*

$$\sum_{i\in[n]} \frac{a_i}{\sum_{j\leq i} a_j} \;\leq\; 1 + \ln \frac{\sum_{i=1}^n a_i}{a_1}$$

Proof. For $n = 1$, the statement is trivially true. For $n \geq 2$, define $b_i = \sum_{j\leq i} a_j$. Then $a_i = b_i - b_{i-1}$ for $i \geq 2$, and hence

$$\sum_{i\in[n]} \frac{a_i}{\sum_{j\leq i} a_j} \;=\; 1 + \sum_{i=2}^{n} \frac{b_i - b_{i-1}}{b_i} \;=\; 1 + \sum_{i=2}^{n} \left(1 - \frac{b_{i-1}}{b_i}\right) \tag{38}$$

Let $r_i = \frac{b_i}{b_{i+1}}$, and let $y = \sum_{i=2}^n (1 - \frac{b_{i-1}}{b_i})$, then $y = \sum_{i=1}^{n-1}(1 - r_i) = (n-1) - \sum_{i=1}^{n-1} r_i$. Since each $r_i \leq 1$ and $\prod_{i=1}^{n-1} r_i = b_1/b_n$, by Lemma 43,

$$y \;\;\leq\;\; (n-1) - (n-1)\left(\frac{b_1}{b_n}\right)^{1/(n-1)}\,. \tag{39}$$

Let $c = \frac{b_1}{b_n}$ and $z = (n-1) - (n-1)c^{1/(n-1)}$. Differentiating z w.r.t. n,

$$\frac{\partial z}{\partial n} \;\;=\;\; 1 - c^{1/(n-1)} + \frac{c^{1/(n-1)}}{n-1} \ln c$$

and again $\partial^2 z/\partial n^2 \;=\; -c^{1/(n-1)} \ln^2 c/(n-1)^3 \;<\; 0$. Hence, z is maximized when $(n-1) - (n-1)c^{1/(n-1)} \;=\; c^{1/(n-1)} \ln \frac{1}{c}$. Substituting the expression on the left in this equality in (39) gives us $y \leq c^{1/(n-1)} \ln \frac{1}{c}$, and since $c = \frac{b_1}{b_n} \leq 1$,

$$y \leq \ln \frac{1}{c} = \ln \frac{b_n}{b_1} = \ln \frac{\sum_{i=1}^n a_i}{a_1}\,. \tag{40}$$

Then from (38) and (40), and by definition of y,

$$\sum_{i\in[n]} \frac{a_i}{\sum_{j\leq i} a_j} \;\leq\; 1 + \ln \frac{\sum_{i=1}^n a_i}{a_1}\,.$$

□

Lemma 45 is used in the proof of Lemma 5 in Section 2. As in Section 2, $\mu := 1 + \frac{1}{3\ln(em)}$.

Lemma 45. *Given* $\mathbf{x}'$ *and* $\mathbf{x}''$ *with* $\tilde{\lambda}(\mathbf{x}') \le 3\ln(em)$ *and* $x'_j \le x''_j \le \mu x'_j$ *where* $\mu = 1 + \frac{1}{3\ln(em)}$, $r_j(\mathbf{x}'') \le e r_j(\mathbf{x}')$.

Proof. By definition of $r(\mathbf{x})$ in (3), and since $x'_j \le x''_j \le \mu x'_j$,

$$r_j(\mathbf{x}'') \;=\; \frac{\sum_{k\in[m]} \tilde{p}_{kj} \exp(\tilde{\mathbf{P}}\mathbf{x}'')_k}{\sum_{k\in[m]} \exp(\tilde{\mathbf{P}}\mathbf{x}'')_k} \;\le\; \frac{\sum_{k\in[m]} \tilde{p}_{kj} \exp(\mu(\tilde{\mathbf{P}}\mathbf{x}')_k)}{\sum_{k\in[m]} \exp(\tilde{\mathbf{P}}\mathbf{x}')_k}\,. \tag{41}$$

Since $\tilde{\lambda}(\mathbf{x}') \le 3\ln(em)$, $\forall k$, $(\tilde{\mathbf{P}}\mathbf{x}')_k \le 3\ln(em)$, and hence $\mu(\tilde{\mathbf{P}}\mathbf{x}')_k = (\tilde{\mathbf{P}}\mathbf{x}')_k + (\tilde{\mathbf{P}}\mathbf{x}')_k/(3\ln(em))$ $\le (\tilde{\mathbf{P}}\mathbf{x}')_k + 1$. Substituting $(\tilde{\mathbf{P}}\mathbf{x}')_k + 1$ for $\mu(\tilde{\mathbf{P}}\mathbf{x}')_k$ in (41) yields

$$r_j(\mathbf{x}'') \;\le\; \frac{\sum_{k\in[m]} \tilde{p}_{kj} \exp((\tilde{\mathbf{P}}\mathbf{x}')_k + 1)}{\sum_{k\in[m]} \exp(\tilde{\mathbf{P}}\mathbf{x}')_k} \;=\; e\,\frac{\sum_{k\in[m]} \tilde{p}_{kj} \exp(\tilde{\mathbf{P}}\mathbf{x}')_k}{\sum_{k\in[m]} \exp(\tilde{\mathbf{P}}\mathbf{x}')_k} \;=\; e\tilde{r}_j(\mathbf{x}')\,,$$

proving the lemma. □

Lemma 46 is used in the proof of Lemma 21 in Section 3. Define $\mu := 1 + \frac{1}{6\ln(emn)}$. We assume that $\mathbf{x}'$, $\mathbf{x}''$ satisfy $x'_{ij} \le x''_{ij} \le \mu x'_{ij}$ for all i, j; that $\mathrm{cost}(\mathbf{x}') \le 6Z\ln(emn)$, and that $\Gamma\tilde{w}_i(\mathbf{x}') = x'_{ij}$ iff $\Gamma\tilde{w}_i(\mathbf{x}'') = x''_{ij}$ for all i, j.

Lemma 46. *For all* i, j, $\mathrm{rate}_{ij}(\mathbf{x}'') \le e\,\mathrm{rate}_{ij}(\mathbf{x}')$.

Proof. Given $\mathbf{x}'$, $\mathbf{x}''$ as defined in the lemma, and a fixed facility i, we will show that

$$\frac{\exp(\sum_{j'} l_{ij'} x''_{ij'}/(Z\Gamma))}{\sum_{i'} \exp(\sum_{j'} l_{i'j'} x''_{i'j'}/(Z\Gamma))} \;\le\; e\,\frac{\exp(\sum_{j'} l_{ij'} x'_{ij'}/(Z\Gamma))}{\sum_{i'} \exp(\sum_{j'} l_{i'j'} x'_{i'j'}/(Z\Gamma))}\,, \tag{42}$$

and

$$\frac{e^{x''_{ij}/\Gamma}}{\sum_{i',j'} e^{x''_{i'j'}/\Gamma}} \;\le\; e\,\frac{e^{x'_{ij}/\Gamma}}{\sum_{i',j'} e^{x'_{i'j'}/\Gamma}}\,. \tag{43}$$

Since the other terms in $\mathrm{rate}_{ij}(\mathbf{x})$ are constants, this will prove the lemma.

Since $Z\tilde{\lambda}(\mathbf{x}') \le \mathrm{cost}(\mathbf{x}')$, and $\mathrm{cost}(\mathbf{x}') \le 6Z\ln(emn)$ by the condition in the lemma statement, $\tilde{\lambda}(\mathbf{x}') \le 6\ln(emn)$. Hence $\sum_{j'} l_{ij'} x'_{ij'}/(Z\Gamma) \le \tilde{\lambda}(\mathbf{x}') \le 6\ln(emn)$, and $x'_{ij}/\Gamma \le \tilde{\lambda}(\mathbf{x}') \le 6\ln(emn)$. Thus

$$\frac{\sum_{j'} l_{ij'} x''_{ij'}}{Z\Gamma} \;\le\; \mu\frac{\sum_{j'} l_{ij'} x'_{ij'}}{Z\Gamma} \;=\; \left(1 + \frac{1}{6\ln(emn)}\right)\frac{\sum_{j'} l_{ij'} x'_{ij'}}{Z\Gamma} \;\le\; \frac{\sum_{j'} l_{ij'} x'_{ij'}}{Z\Gamma} + 1 \tag{44}$$

and

$$\frac{x''_{ij}}{\Gamma} \;\le\; \mu\frac{x'_{ij}}{\Gamma} \;=\; \left(1 + \frac{1}{6\ln(emn)}\right)\frac{x'_{ij}}{\Gamma} \;\le\; \frac{x'_{ij}}{\Gamma} + 1\,. \tag{45}$$

Then (42) and (43) follow from (44) and (45) respectively. □

For the next lemma, we are given $\mathbf{p}, \mathbf{x}$ and $\mathbf{u} \in \mathbb{R}^n_+$, with the elements of $\mathbf{u}$ non-decreasing. For $1 \le k \le n$, define

$$T_k \;:=\; \frac{1}{u_k}\sum_{j=1}^{k} p_j x_j$$

and $T := \max_k T_k$. We also define

$$P \;:=\; \sum_{j=1}^{n} \frac{p_j x_j}{u_j}\,.$$

Lemma 47. *With P, T and* $\mathbf{u}$ *defined as above,* $P \;\le\; T\left(1+\ln\frac{u_n}{u_1}\right)$.

Proof. We first obtain a different expression for P, and then relate the expression we obtain to T.

$$\begin{aligned}
P \;&=\; \sum_{j=1}^{n} \frac{p_j x_j}{u_j} \;=\; \sum_{j=1}^{n} \frac{p_j x_j}{u_n} + \sum_{j=1}^{n} p_j x_j \left(\frac{1}{u_j} - \frac{1}{u_n}\right) \\
&=\; T_n + \sum_{j=1}^{n} p_j x_j \sum_{k=j}^{n-1} \left(\frac{1}{u_k} - \frac{1}{u_{k+1}}\right) \;=\; T_n + \sum_{k=1}^{n-1}\sum_{j=1}^{k} p_j x_j \left(\frac{1}{u_k} - \frac{1}{u_{k+1}}\right) \\
&=\; T_n + \sum_{k=1}^{n-1} \left(1 - \frac{u_k}{u_{k+1}}\right) \frac{1}{u_k} \sum_{j=1}^{k} p_j x_j \;=\; T_n + \sum_{k=1}^{n-1} \left(1 - \frac{u_k}{u_{k+1}}\right) T_k \\
&\le\; T\left(1 + \sum_{k=1}^{n-1} \left(1 - \frac{u_k}{u_{k+1}}\right)\right) \qquad (46)
\end{aligned}$$

The expression on the right in (46) is exactly the same as the expression on the right in (38), with $b_{i-1} = u_k$. Then from (40), since $c = u_1/u_n$ and $y = \sum_{k=1}^{n-1}\left(1 - \frac{u_k}{u_{k+1}}\right)$,

$$P \;\le\; T\left(1 + \ln\frac{u_n}{u_1}\right)\,.$$

□